OULUN SEUDUN
AMMATTIKORKEAKOULU

Ilpo Siira

ENERGIANKERÄYSMENETELMIEN KARTOITUS

ENERGIANKERÄYSMENETELMIEN KARTOITUS

Ilpo Siira
Opinnäytetyö
Syksy 2012
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikkan koulutusohjelma, langaton tietoliikenne

Tekijä(t):Ilpo Siira
Opinnäytetyön nimi: Energiankeräysmenetelmien kartoitus
Työn ohjaaja(t): Henry Hinkula ja Kari Jyrkkä
Työn valmistumislukukausi ja -vuosi: Syksy 2012 Sivumäärä: 32 + 2 liitettä

Työn toimeksiantajana oli Oulun seudun ammattikorkeakoulun Tekniikan yksikkö ja työn tarkoituksena oli energiankeräysmenetelmien kartoittaminen ja testaaminen. Työllä oli kolme tavoitetta: energiankeräysmenetelmien kartoittaminen, demolaitteiston rakentaminen energiankeräysmenetelmän tastausta varten ja 1–2 sivun mittainen kooste vertailusta englanniksi.

Energiankeräymenetelmien kartoitus tehtiin Internetistä saatujen lähteiden perusteella ja demolaitteistoa varten tehty vertailu tehtiin etsimällä testausalustoja Internetistä valmistajien ja käymällä jälleenmyyjien sivuilla. Demojärjestelmää varten tilatut laitteet valittiin työtä varten kirjoitetun vertailudokumentin perusteella. Demojärjestelmään tilatut laitteet pystyivät keräämään energiaa valosta ja tärinästä, ja molempia keräysmenetelmiä testattiin erikseen. Mittauksissa kokeiltiin valitun testausalustan valokennon toimintaa eri valonlähteillä mittaamalla CHARGE-pinnistä jännitettä, josta voidaan nähdä, tuottaako velokenno tarpeeksi jännitettä piirilevyn akkujen lataamiseen. Valmistaja ei ollut virittänyt pietsosähkönkerääjiä ja ne täytyi virittää, jotta niillä saatiin tuotettua mahdollisimman suuri jännite. Pietsosähköenergian kerääjät viritettiin täristimen avulla 100 Hz:n taajuudelle lisäämällä pietsoelementin toiseen päähän painoa.

Testausalustassa oleva valokenno toimi kaikilla testatuilla valoilla ja pietsosähkönkerääjän virittäminen 100 Hz:n taajuudelle vaati 0,26 gramman painon lisäystä, jolloin sen tuottama jännite oli 39,1 mV. Valosta saatava jännite riittää mikrokontrollerin päällä pitämiseen, langattoman yhteyden ylläpitämiseen, datan lähetykseen ja lämpötila-anturin mittauksiin. Pietsosähkönkerääjän tuottaman jännite riittää sensorille ja mikrokontrollerille, jos tärinä on tarpeeksi voimakasta. Pietsosähkönkerääjä, johon on yhdistetty valokenno, on tarpeeksi tehokkas tuottamaan demojärjestelmän vaatiman jännitteen.

Asiasanat: uusiutuvat energialähteet, valokennot, liike-energia, akut

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology
Option of Wireless Devices and Networks

Author(s): Ilpo Siira
Title of thesis: Study of energyharvesting methods
Supervisor(s): Henry Hinkula and Kari Jyrkkä
Term and year when the thesis was submitted: Autumn 2012 Pages: 32 + 2 appendices

---

The assigner for the thesis was Oulu University of Applied Sciences School of Engineering and the main goal for the thesis was to study and test energy harvesting methods. The thesis had three objectives. To study energy harvesting methods, to build a demo system for testing the energyharvesting methods and to write a summary of two page length of the comparison done on development kits.

The study of energy harvesting methods was done with the help of Internet and the comparison of the energy harvesting kits was done by visiting the manufacturers and retailers sites. The devices that were ordered for the demo system were based on from the comparison document. In the measurements the photovoltaic cell in the development kit was tested by measuring the voltage of the CHARGE pin. Wether or not the photovoltaic cell is charging the development boards energy chips, can be seen from the charge pin. The piezoenergyharvesters were not tuned by the manufacturer and they had to be tuned to ensure the highest possible voltage that the device can produce. The piezoenergyharvesters were tuned to a frequency of 100 Hz with a shaker by adding weight to the other end of the piezoelement.

The photovoltaic cell in the development kit worked with every light source used in the tests and the tuning of the piezoelement to 100 Hz needed 0.26 grams of added weight. With the added weight the piezoelement produced a voltage of 39.1 mV. The voltage harvested from the light was enough to power the microcontroller, maintain the wireless connection, sending data and for the measurements done by the temperature sensor. The piezoelectric harvester produced at least enough voltage for the sensor and the microcontroller if the shaking is strong enough. Piezoelectric harvester combined with a photovoltaic cell was efficient enough for the demo system.

Keywords: regenerative sources of energy, photocells, kinetic energy, batteries

SISÄLLYS

1 JOHDANTO

Tämä opinnäytetyö käsittelee energiankeräysmenetelmien kartoitusta ja demolaitteiston rakentamista mikrokontrollerialustalle. Työn tilaaja oli Oulun seudun ammattikorkeakoulun Tekniikan yksikkö. Energiankeräysmenetelmistä kartoitettiin menetelmät, joilla voi tuottaa energiaa pienille laitteille, kuten esimerkiksi ihmisen kehon toimintaa seuraaville laitteille.

Ensisijainen tavoite oli käydä läpi erilaiset energiankeräysmenetelmät ja suorittaa niiden välistä vertailua. Tähän työhön valittavaa tekniikkaa käytetään älyvaatteissa olevien laitteiden käyttöiän pidentämiseen. Vertailun tarkoituksena oli, että sen avulla voidaan valita parhaiten sopiva menetelmä myös muihin energiankeräystä käyttäviin projekteihin. Vertailusta kirjoitettiin myös 1–2 sivun kooste tilaajalle englanniksi. Kirjoitettua vertailua voidaan käyttää apuna myöhemmin tehtävissä projekteissa, joissa käytetään energiankeräysmenetelmiä. Menetelmän valinnan jälkeen vertailtiin valittuun energiankeräysmenetelmään eri valmistajilta löytyviä testausalustoja. Testausalustan valinnan jälkeen rakennettiin demolaitteisto.

Työn tarkoituksena oli tutkia mahdollisuutta saada tuotettua älyvaatteiden vaatima jännite energiankeräysmenetelmien avulla ja tutkia, voidaanko jännitettä tuottaa tarpeeksi itsenäiseen laitteeseen vai käytetäänkö energiankeräystä akkujen lataamiseen. Älyvaatteet voivat esimerkiksi tarkkailla ihmisen elintoimintoja, kuten sydäntä. Kerätyn jännitteen tulisi riittää langattomien yhteyksien ylläpitämiseen, datan lähetykseen sekä sensorien ja mahdollisesti mikrokontrollerien käyttämän jännitteen tarpeeseen.

2 ENERGIANKERÄYSMENETELMÄT

Sensoreille tarkoitettuja energiankeräysmenetelmiä on olemassa useampia, ja niiden tuottama jännite ei ole tasaista. Jännitteen saannin suhteen eri menetelmillä on eroja. Niiden keräämä jännite ilmoitetaan yleensä milliwatteina (mW) tai mikrowatteina (µW). Kontrolleripiirin tarvitsemissa jännitteissä käytetään esimerkkinä Arduino Duemilanove -kontrolleripiiriä. Arduino Duemilanove -kontrolleripiirissä on 5 V:n käyttöjännite. Mahdollista on käyttää myös 3,3 V:n pinnejä, jolloin käyttöjännite on 3,3 V. Tulojännitteen suositus on 7–12 V ja tulojännitteen rajat ovat 6–20 V DC. Virta I/O-pinniä kohden on 40 mA ja 3,3 V:n pinniä kohden 50 mA. (1.)

Arduino ei ole tässä tapauksessa paras vaihtoehto mikrokontrolleripiiriksi, koska on olemassa virtapihimpiäkin mikrokontrolleripiirejä. Demojärjestelmässä käytettyssä MSP430-mikrokontrolleripiirissä on käyttöjännitteenä 1,8 V tai 3,6 V. MSP430 kuluttaa virtaa aktiivisessa tilassa tyypillisesti 270 µA ja maksimissaan 390 µA. Valmiustilassa virrankulutus on tyypillisesti 0,7 µA ja maksimissaan 1,4 µA.

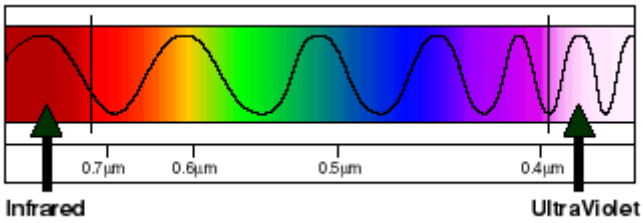
Käyttötarkoitus ja olosuhteet tulee ottaa huomioon menetelmää valittaessa. Esimerkiksi valokennoa ei kannata käyttää, jos sitä ei voida asentaa paikkaan, jossa valokenon pinnalle ei muodostuu varjoja. Tällä hetkellä käytettävissä olevia menetelmiä ovat valo, värinä, lämpö ja radiotaajuusenergiankeräys.

2.1 Valo

Valosta saadaan energiaa kahdella järjestelmällä. Nämä järjestelmät ovat fotosähkö- ja aurinkoenergiajärjestelmä. Fotosähkökennoilla voidaan tuottaa sähköä myös sisätiloissa löytyvillä loistevalaisimilla. Valosta saadaan energiaa joko varastoimalla valosta eri valon allonpituuksia eli värejä (kuva 1) tai keräämällä energiaa suoraan auringon tuottamasta lämmöstä. Aurinkoenergian ja fotosähkön ero on, että aurinkoenergia käyttää linssejä ja peilejä, joilla auringon valo keskitetään fotosähköpaneeleihin, tai niitä käytetään lämmönlähteenä. Tästä syystä aurinkolämpöenergiaa käytetään pelkästään laajamittaisessa energiantuotannossa ja fotosähkö sopii paremmin pienemmille järjestelmille. (2.)

Näkyvällä valolla on seuraavat aallonpituusalueet:

- violetti 380–450 nm
- sininen 450–490 nm
- vihreä 490–560 nm
- keltainen 560–590 nm
- oranssi 590–630 nm
- punainen 630–760 nm. (3.)



KUVA 1. Näkyvän valon aallonpituudet (4)

Fotosähkö- ja aurinkoenergiajärjestelmiä on monenlaisia. Tästä ovat esimerkkeinä aurinkoparistot, aurinkoparistolaturit ja aurinkopaneelit. Aurinkoparisto on suljettu ja huoltovapaa laite, joka tarvittaessa varaa itseensä aurinkokennon keräämää energiaa. Aurinkopatterilatureilla voidaan ladata laitteiden akkuja johdon avulla (kuva 2). Käytettävä johto voi olla esimerkiksi USB-johto. Aurinkopaneeli on laite, joka muuttaa energian käytettävään muotoon, joko suoraan auringon lämmöstä tai muuttamalla valon jännitteeksi. Fotosähköisessä järjestelmässä voi olla varapatterit tai katkoton virtalähde (UPS), jotka voivat pitää valittuja piirejä käytössä tunteja tai jopa päiviä. (2.)

KUVA 2. Fotosähkölaturi (6)

Fotosähkö- ja aurinkoenergiajärjestelmillä on erilaiset komponentit ja tekniset tiedot. Fotosähköisessä järjestelmästä löytyviä komponentteja ovat esimerkiksi DC-AC-vaihtosuuntain ja avustavat energialähteet. Fotosähkölaturi käyttää pulssileveysmodulaatiota ja kolmivaihelatausmenetelmiä. (2.)

Fotosähkössä käytettävissä kennoissa on yleensä yksi piiseos ja niiden hyötysuhde on suunnilleen 15 %. Piiseokset säädetään varastoimaan vain tiettyä valon taajuutta. Piiseosten lisäämisellä kerroksittain voidaan varastoida valoa useammilta taajuuksilta. Paneelin lämpötila voi kesällä saavuttaa 50 ˚C:n lämpötilan, jolloin teho laskee 12 % 25 ˚C:n lämpötilaan verrattuna. (7.)

Paneelit pitää sijoittaa siten, että niihin ei tule varjoja, jotka laskisivat paneelin tehoa. Jo pienellä varjolla paneelissa voi olla merkittävä merkitys jännitteen tuottamiseen. Jos paneelit on asennettu sarjaan, myös muiden paneelien teho laskee yhden paneelin varjon takia. (7.)

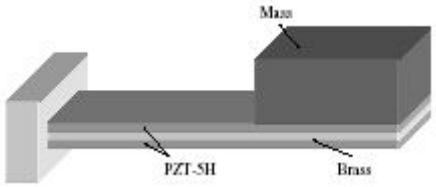
2.2 Värinä

Värinällä voidaan tuottaa sähköä kolmella eri tavalla: pietsosähköinen, sähkömagneettinen ja sähköstaattinen menetelmä. Pietsosähköisessä muuntimessa värinä vääristää sensorin pietsoelementtistä materiaalia, jolloin se muodostaa jännitettä. Pietsosähköisissä materiaaleissa esiintyy tyypillisesti anisotrooppisia ominaispiirteitä, mistä johtuen materiaalien ominaisuudet eroavat voiman suunnan sekä polarisaation ja elektrodien orientaation mukaan. (8, s. 5.)

Sähkömagneettisissa muuntimissa käytetään Faradayn keksimää sähköinduktiota, jossa käämin sisällä liikkuva magneetti aiheuttaa muutoksia magneettikentässä. Tällöin magneettikentässä olevaan sähköjohtimeen muodostuu virta. Sähköstaattisessa muuntimessa kondensaattorin jännite pidetään tasaisena. Kondensaattorin levyjen etäisyyden kasvaessa kapasitanssi pienenee ja kondensaattorista purkautuu varaus. Levyjen ja varausten liikuttamiseen tarvittava mekaaninen energia kerätään ja varastoidaan toiseen kondensaattoriin tai ladattaviin pattereihin. (9, s. 1, 2.)

2.2.1 Pietsosähkö

Pietsosähköisen materiaalin positiivinen ja negatiivinen sähkölataus on erotettu toisistaan ja ne on hajautettu tasaisesti siten, että materiaali on kokonaisuudessaan sähköisesti neutraali. Rasituksessa tasasuhtaisuus hajoaa ja tästä syntyy jännite (kuva 3). Pietsosähköinen vaikutus esiintyy vain sähköä johtamattomissa materiaaleissa, jotka voidaan jakaa kahteen ryhmään: kristallit ja keraamiset materiaalit. Parhaiten tunnettu pietsosähköinen materiaali on kvartsi (15). Kuvassa 3 on ulokepalkillinen pietsosähköinen generaattori, joka heiluu värinän mukana. Päähän lisätyllä painolla saadaan säädettyä tuotetun jännitteen määrä tietyllä taajuudella mahdollisimman suureksi.



KUVA 3. Ulokepalkillinen pietsosähköinen generaattori (8, s. 9)

2.2.2 Sähkömagneettinen energiankeräys

Faradayn lain mukaan kaikki muutokset kelan magneettikentässä muodostavat jännitteen, joka on suhteessa käämien lukumäärään, pinta-alaan, avaruudellisesti muuttuvan magneettikentän nopeuden ja muuttuvan magneettikentän ajan itseisarvon suhteen. Faradayn induktiolaki lasketaan kaavoilla 1 ja 2 (10, s. 2).

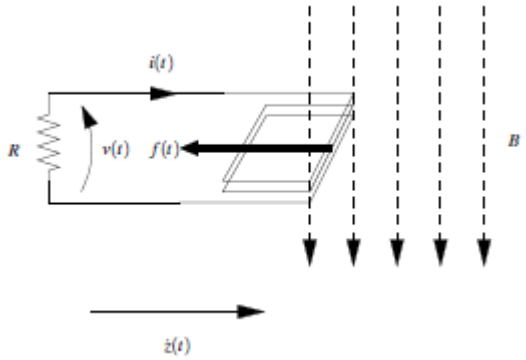
$$\oint_l E * dl = -\frac{d}{dt}\int_s B * ds = -\frac{d\varphi_m}{dt} \qquad \textit{KAAVA 1}$$

$$V(t) = N2\pi f B_0(x,t)A \qquad \textit{KAAVA 2}$$

N = kelan kierrosten lukumäärä
A = kelan poikkipinta-ala
2πf = avaruuskulma
B_0 = muuttuvan magneettikentän ajan itseisarvo

Jännitteen nostamiseksi mitä tahansa yllä olevan kaavan suureista voidaan nostaa, mutta liian suuret muutokset laskevat suorituksen tehoa. Kelan kierros-

ten lukumäärän nostaminen lisää vastusta ja se laskee tehoa. Myös kelan muoto vaikuttaa tehoon, koska ajan mukaan muuttuvan magneettikentän kaltevuuden täytyy olla koottu kelan paksuuden läpi. Joissakin tapauksissa magneettivuo voi purkautua ja laskea tehoa. Värinän lisääminen lisää jännitettä, mutta pienentää liikkuvan elementin siirtymää magneettikentässä. Kuva 4 esittää elektromagneettisen anturin toimintaa. (10.)

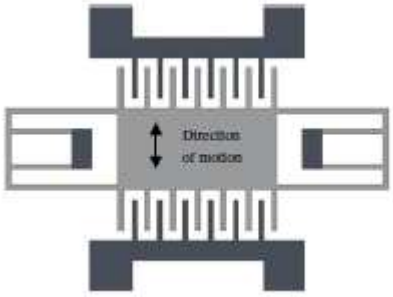


KUVA 4. Elektromagneettisen anturin toimintaperiaate (11, s. 23)

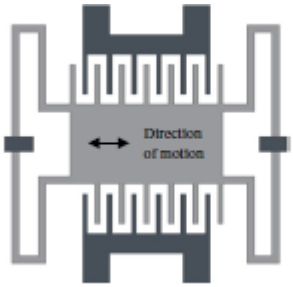
2.2.3 Sähköstaattinen energiankeräys

Sähköstaattista energiaa saadaan muodostamalla värinää sähkökenttään, joka on riippuvainen levykondensaattorin vaihtuvasta kapasitanssista. Värinä liikuttaa varatun kondensaattorin toista levyä. Värinän vaikutuksesta levyjen välinen etäisyys muuttuu, mikä aiheuttaa kondensaattorin levyjen yli olevan jännitteen muuttumisen. (9.)

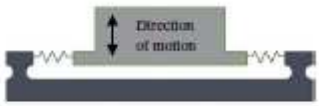
Kondensaattorin pitää olla ladattu ennalta täyteen varaukseen, jossa levyjen etäisyys on mahdollisimman pieni, ennen kuin jännitettä voidaan kerätä. Ennalta ladattu jännite tarvitaan keräyskierron aloittamiseksi, ja aloittamiseen käytetyn jännitteen täytyy olla vain murto-osa saadusta jännitteestä. Kuvat 5, 6 ja 7 esitettävät MEMS-kondensaattorikeräimen toimintaperiaatteen, jossa on nuolilla merkitty levyn liike, jolla jännitettä tuotetaan. (9.)



KUVA 5. Pintojen välissä päällekäisyys vaihtelee (8, s. 14)



KUVA 6. Pintojen välissä aukko sulkeutuu (8, s. 14)



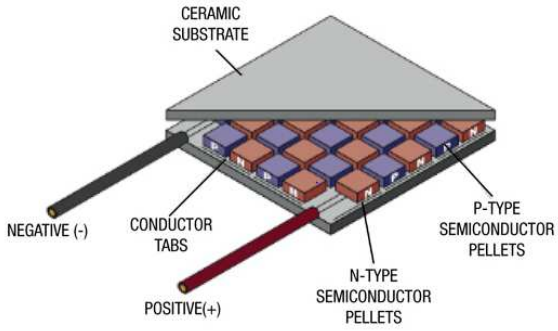
KUVA 7. Tasossa oleva aukko sulkeutuu (8, s. 14)

2.3 Lämpöenergian keräys

Lämpöenergian keräysjärjestelmä hyödyntää sen kahden pinnan välistä lämpötilaeroa. Jo ihmisen ihon ja normaalilämpöisen huoneen välinen lämpötilaero on noin 10 ˚C, mutta ihmisestä haihtuva lämpö on merkityksetöntä verrattuna nykyajan laitteisiin. (12.)

Sillä, mitataanko muutaman asteen vai 100 ˚C:n lämpötilaeroja, ei ole merkitystä, sillä molemmilla toimii sama toimintaperiaate. Pääideana on lämpöerojännityksen ylläpitäminen. Moduulin ylä- ja alaosa on yleensä keraamista alumiinioksidia. Näiden välissä on n- ja p-tyypin puolijohteita ja moduulin jalat ovat yleensä vismunttitelluridiä tai antimonitelluridiä. Jalat on kinnitetty yhteen muodostamaan sarjaan sähköisen kytkennän ja rinnaikkain lämpökytkennän. Ilman läm-

pötilaeroa ei ole lämpövirtausta eikä myöskään sähköistä ulostuloa. Laitteessa liikkuva lämpövirtaus lämmittää laitetta ja tasaa pintojen lämpötilaeron, ellei toista pinnoista jäähdytetä jollakin tavalla. Kuvassa 8 oleva laite tuottaa vaihtojännitettä pintojen lämpötilaerosta. (12.)



KUVA 8. Lämpöenergiageneraattorimoduuli. (12)

2.4 Radiotaajuusenergiankeräys

Radiotaajuusenergiaa eli RF-energiaa tuottavat lähteet, jotka muodostavat suuren sähkömagneettisen kentän, kuten TV-signaali, langattomat radioverkot ja matkapuhelinverkon antennit. Vastaanottavaan antenniin on kiinnitetty jännitettä tuottava piiri, joka kaappaa radiosignaalia ja muuttaa sen tasajänniteeksi. Yleisimmin radiotaajuusenergiankeräystä käytetään radiotaajuustunnistimissa, jossa RFID-tunnisteen lukija lähettää radiotaajuutta keräyslaitteeseen. Keräyslaite syöttää juuri tarpeeksi energiaa lähettääkseen takaisin tunnistustiedon ja muuttaa lopun energian jänniteeksi. (13.)

Suurin osa piireistä käyttää kelluvaa hilatransistoria diodina, joka muuttaa vastaanotetun signaalin jännitteeksi. Jos laite vaatii suurempaa jännitettä, voidaan transistorin virran kulutukseen liittää kondensaattori ja toinen kelluva hilatransistori, johon on myös kytketty kondensaattori mahdollistamaan suuremman ulostulojännitteen tuottamisen kondensaattorien saavutettua täyden potentiaalinsa. (13.)

Useimmat alat käyttävät tekniikka syrjäisemmillä, vaarallisemmilla tai herkemmillä alueilla, missä tarvitaan huoltovapaata tehoa pienillä jännitteillä, mutta tekniikkaa voidaan käyttää myös mikrokontrollereiden ja sensoreiden vaatiman

energian tuottamiseen. Radiotaajuuden keräämisen huonompana puolena on se, että radiotaajuuden lähteen pitää olla suhteellisen lähellä energiankeräyskohdetta, jotta kaikki mahdollinen signaalin teho saadaan kerättyä laitteen käyttöön.

Taulukkoon 1 on merkitty eri energiankeräysmenetelmillä saatavia tehoja lähteen tehon mukaan. Taulukosta voidaan päätellä, minkälaisia tehoja milläkin energiankeräysmenetelmällä voidaan saada.

TAULUKKO 1. Energiankeräysmenetelmillä saatavilla oleva teho (14)

Lähde		**Lähteen teho**	**Kerätty teho**
Valo	Sisällä (loistelamppu)	0,1 mW/cm^2	10 µW/cm^2
	Ulkona (aurinko)	100 mW/cm^2	10 mW/cm^2
Värinä/liike	Ihminen	0,5 m 1 Hz:llä* 1 m/s^2 50 Hz:llä*	4 µW/cm^2
	Laite	1 m 5 Hz:llä* 10 m/s^2 1 kHz:llä*	100 µW/cm^2
Lämpö	Ihminen	20 mW/cm^2	30 µW/cm^2
	Laite	100 mW/cm^2	1–10 mW/cm^2
Radiotaajuus	GSM-tukiasemajärjestelmä	0,3 µW/cm^2	0,1 µW/cm^2

* Värinän taajuus ja voimakkuus.

3 TESTAUSALUSTOJEN VERTAILU

Työssä verrattiin testaus- ja demoalustoja sekä muutamaa energiankeräyslaitetta. Vertailun avulla valittiin laitteistot, joilla demojärjestelmä rakennettiin. Vertailusta kirjoitettiin myös vertailudokumentti (liite 1). Dokumentti on kirjoitettu englanniksi ja sitä on mahdollista käyttää muidenkin energiankeräysprojektien laitevertailun tekemiseen.

Vertailussa otettiin huomioon laitteiden hinta, saatavuus, laitteen käyttämä energiankeräysmenetelmä, mahdollisuus liittää muita energiankerääjiä ja valmistajan ilmoittama luku kerätyn energian määrästä. Vertailun tuloksena demojärjestelmään valittiin Volturen PEH 25w- ja SEH 25w -energiankerääjät. Volturen PEH 25w (kuva 9) ja SEH 25w (kuva 10) ovat pietsosähkönkerääjiä, eli ne keräävät energiaa värinästä ja SEH 25w:ssä on myös valokenno. Molemmat Volturen energiankerääjät tilattiin virittämättöminä versioina, joten ne täytyi virittää toimimaan tietyllä taajuudella.

KUVA 9. Volture PEH 25w

KUVA 10. Volture SEH 25w

Testausalustaksi valittiin Texas Instrumentin eZ430-RF2500-SEH Solar Energy Harvesting Development Tool, jossa on Cymbetin msp430-mikrokontrolleripiiri. EZ430-RF2500-SEH toimii valokennon tuottamalla energialla ja siihen voi liittää oman energiankerääjän. Testausalustassa tuli myös valmis demo-ohjelma, joka on koodattu C++-ohjelmointikielellä. Valmistaja ilmoitti koodin olevan vapaata lähdekoodia sekä vapaasti muokattavissa. EZ430-RF2500-SEH Solar Energy Harvesting Development Toolin mukana tulevat laitteet ovat kuvissa 11, 12 ja 13.

KUVA 11. EZ430-RF2500-SEH

KUVA 12. EZ430-RF2500T kiinnitettynä pattereihin

KUVA 13. Tietokoneen USB-porttiin tuleva eZ430-RF2500T

Testausalustassa oleva valokenno vaatii toimiakseen 200 luxin valaistuksen, ja täyden varaustehon se saavuttaa 700 luxissa. Testausalustan akkuina toimii kaksi CBC050-M8C enerchipiä. Enerchip on puolijohdeakku. Yksi CBC050-M8C enerchip varaa virtaa 50 µAh.

4 DEMOJÄRJESTELMÄ

Demojärjestelmää varten tilatussa eZ430-RF2500-SEH-testausalussa on valokenno, jonka on tarkoitus tuottaa tarpeeksi suuri jännite suorittamaan seuraavia komponentteja: piirilevyä, johon valokenno on kiinnitetty, lämpötila-anturia ja langatonta yhteyttä tietokoneeseen USB-liitännällä tulevaan päätelaitteeseen.

Useammalla testausalustalla voi myös tehdä testausalustojen välille oman verkon, jossa testausalustat toimivat päätelaitteina ja lähettävät datan tietokoneelle, missä dataa voidaan seurata demo-ohjelmasta löytyvästä käyttöliittymästä. Testausalustojen lähettämä data sisältää lämpötilan, käyttöjännitteen ja langattoman yhteyden vahvuuden. Data sisältää myös tiedon siitä, lataako piirilevy akkuja, ja jos ei lataa, monenko viestin lähettämiseen akuissa oleva virta riittää.

Demoamisen aikana oli tarkoitus tutustua valmiiseen demo-ohjelman sisältöön ja muokattavuuteen (liite 2), testata testausalustan valokennoa eri valonlähteillä ja arvioida testattujen valonlähteiden aallonpituudet. Lisäksi tuli varmistaa valokennon tuottaman jännitteen riittävyys testausjärjestelmälle, selvittää testausalustan virran kesto akkujen ollessa täynnä ja kun niitä ei ladata sekä pohtia, voiko eZ430-RF2500-SEH testausalustaa käyttää opetuksessa. Vielä tehtävänä oli Volture PEH 25w:n ja SEH 25w:n energiankerääjien virittäminen ja Volturen laitteiden muodostaman jännitteen määrän selvittäminen.

Testauksessa oli käytetty seuraavia laitteita:

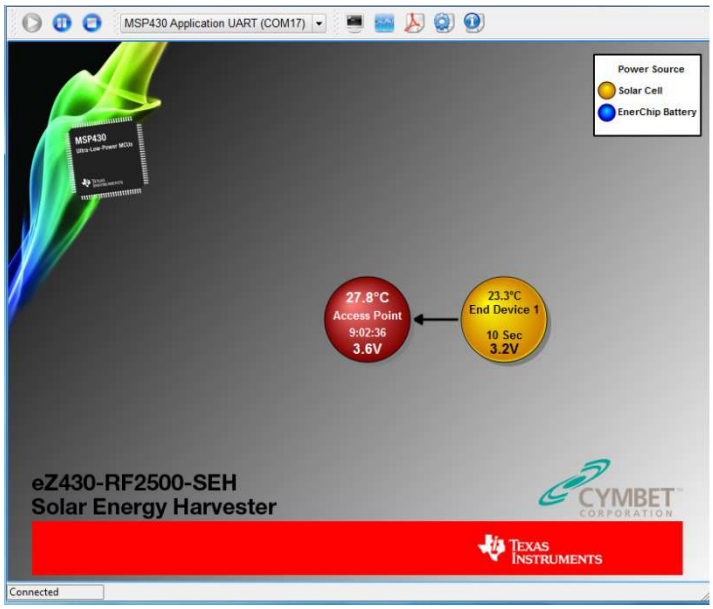
- Dactron Comet Shaker -tärinäalusta
- lux-mittari
- yleismittari
- tärinäalustaan tuleva kiinnityslevy.

4.1 Demo-ohjelma

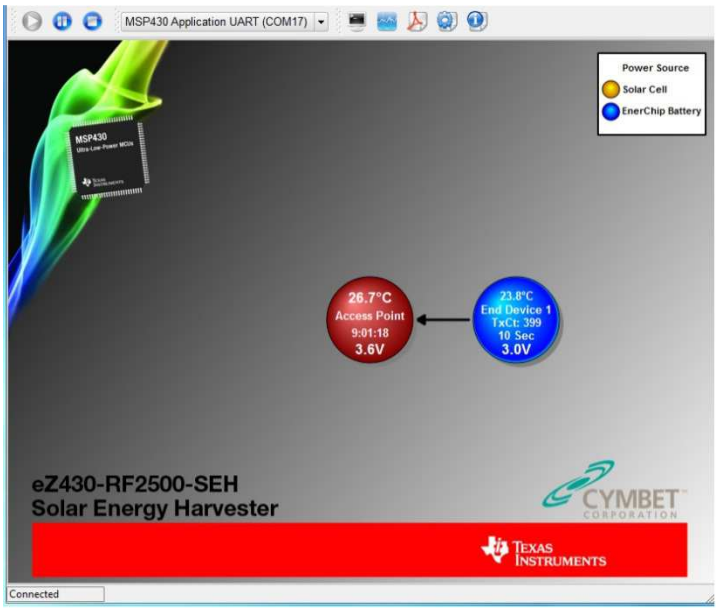
Valmis demo-ohjelma on ohjelmoitu C++-ohjelmointikielellä ja se on koodattu QT-ohjelmalla QWT-kirjastoa käyttäen. Koodi on käännetty C++ 2008 Express Editionilla. Käytetyt ohjelmat ja kirjastot ovat ilmaisia ja kaikille saatavissa.

Demo-ohjelma on jaettu aliohjelmiin, joissa jokaisella päätoiminnolla on oma aliohjelmansa. Tämän takia koodi on helppo ymmärtää, vaikka ei osaisikaan kunnolla C++-ohjelmointia. Koodi on myös vapaasti muokattavissa ja sitä voi myös käyttää pohjana omalle käyttöjärjestelmälle.

Demo-ohjelmaan kuuluu tietokoneella näkyvä käyttöliittymä, joka näyttää testausalustojen käyttöjännitteen sekä lämpötila-anturin mittaaman arvon celsiuksina tai fahrenheiteina. Käyttöliittymä näyttää myös, mitkä laitteet eivät lataa akkuja ja montako datalähetystä ne voivat lähettää ennen virran loppumista. Käyttöliittymässä voi myös tarkkailla saapuneita paketteja konsoli-ikkunan avulla, josta voi nähdä myös langattoman yhteyden vahvuuden. Demo-ohjelman käyttöliittymä on kuvissa 14 ja 15.



KUVA 14. Demo-ohjelman käyttöliittymä, kun valokenno on käytössä



KUVA 15. Demo-ohjelman käyttöliittymä, kun valokenno ei ole käytössä

4.2 Testausalustan valokennon testaus

Valokennoa testattiin sisätiloissa ja ulkona. Ulkona testaukset suoritettiin loppukeväällä vesisateen aikana ja suoraan auringosta. Sisätiloissa valokennoa testattiin loistelampulla, 40 ja 60 W:n hehkulampulla, 40 W:n halogeenilampulla, 25 W:n energiansäästölampulla ja sinisellä LED-valaisimella, jossa oli 24 sinistä lediä.

Suurin osa sisätiloissa käytetyistä lampuista valaisee huonetta valkoisella valolla, jossa on useita valon aallonpituuksia. Ainoa testauksessa käytetty lamppu, joka ei valaissut valkoisella valolla, on LED-valaisin. Aallonpituudet arvioitiin käyttäen piilasista prismaa, josta valoa heijastamalla voidaan nähdä valon aallonpituuksien värit. Tästä syystä eri valonlähteistä on prisman avulla arvioitu vahvin aallonpituus ja vain vahvin aallonpituus on ilmoitettu. Vahvimman aallonpituuden tunnistaa prismasta värien voimakkuuden perusteella, sillä voimakkaimmalla aallonpituudella on kirkkain ja selkeimmin nähtävä väri. Valon aallonpituuden arvioissa voi olla noin 10 nm:n virhe.

Valokennon toimivuutta tarkkailtiin mittaamalla CHARGE-pinniä yleismittarilla. Kun valokenno tuottaa energiaa ja lataa piirissä olevia enerchipejä, on CHAR-

GE-pinni 0-tilassa ja antaa 0 V:n jännitteen. Kun valokenno ei lataa enerchipejä, CHARGE-pinni on 1-tilassa ja pinnin antama jännite on 1,8 V.

Ensimmäisena testattavana lamppuna oli loistelamppu. Testauksessa käytetyssä tilassa oli useita loistelamppuja päällä ja niiden valaistus oli 460 luxia. Valon aallonpituus oli noin 630 nm ja valaistus oli riittävä testausalustan toimivuuden takaamiseksi.

Seuraavana testattavana lamppuna oli 25 W:n energiansäästölamppu. Lampun valaistus laski nopeasti etäisyyden kasvaessa ja mittauksessa käytetty etäisyys oli 30 cm, jolloin valaistus oli noin 300 luxia. Valon aallonpituus oli noin 600 nm ja energiansäästölamppu pystyi lataamaan testausalustan akkuja.

Sinisen LED-valaistimen valon voimakkuus oli todella pieni ja valokenno piti sijoittaa 20 cm:n päähän valaistimesta, jolloin valon voimakkuus oli 220 luxia. Valon aallonpituus oli noin 470 nm ja riitti testausalustan akkujen lataamiseen. Testauksen aikana testausalustan lämpötila-anturi ja langaton yhteys eivät olleet käytössä.

40 W:n hehkulampulla mittauksessa käytetty etäisyys oli 50 cm ja valaistuksen voimakkuus noin 350 luxia. Valon aallonpituudeksi mitattiin noin 500 nm ja valaistus oli riittävä akkujen lataamiseen. 60 W:n hehkulampulla etäisyys oli metri, valon aallonpituus 600 nm ja valon voimakkuus noin 370 luxia, jolla alustan akkuja saatiin ladattua.

40 W:n halogeenilampulla mittauksessa käytetty etäisyys oli 1,5 metriä ja valaistuksen voimakkuus noin 1900 luxia. Valon aallonpituudeksi saatiin 550 nm ja valokenno latasi piirilevyllä olevia akkuja.

Auringonvalosta suoritetuissa mittauksissa testausalusta sijoitettiin aurinkoisella säällä ulos auringonpaisteeseen ja sateella mittaukset suoritettiin sisätiloissa ikkunasta tulevalla valaistuksella ilman muuta valaistusta. Sateella sisätiloihin ulkoa tuleva valaistus oli kevään lopulla noin 370 luxia. Valokenno latasi akkuja kummassakin tapauksessa.

Valokennon toimintaa testattiin myös silloin, kun testausalusta oli käytössä. Testauksen tarkoituksena oli tarkistaa, pystyykö valokenno tuottamaan tarpeeksi jännitettä mikrokontrolleripiirille, lämpötila-anturille, langattoman yhteyden ylläpitämiselle ja datan lähetykselle. Testaus suoritettiin loistelamppujen testauksen yhteydessä demo-ohjelman käyttöliittymän avulla. Testausalustan akut ladattiin ensin täyteen ja sen jälkeen muodostettiin langaton yhteys testausalustan ja tietokoneen välille. Testausalustan valokennon pitää tuottaa energiaa langattoman yhteyden muodostamisessa, koska yhteyden muodostamisen käyttämä energia voi tyhjentää testausalustan akut kerralla.

Testausalustan annettiin ensin toimia itsestään kymmenen minuuttia, jonka jälkeen valokenno peitettiin. Tietokoneella oleva käyttöliittymä näytti jäljellä olevien datakähetysten määrän akkujen varauksen perusteella ja viiden minuutin päästä valokennon peite otettiin pois. Muutaman minuutin päästä valokenno peitettiin uudelleen ja tarkistettiin nykyisellä virralla tulevien datalähetysten määrä. Tämä toimenpide toistettiin useita kertoja, ja jokaisella kerralla testausalustan akut olivat täynnä valokennon peittämisen jälkeen.

4.3 Testausalustan virran keston selvittäminen

Testausalustan käyttämästä virrasta suurin osa kuluu langattoman yhteyden muodostamiseen ja datan lähetykseen. Datan lähetystiheyttä voidaan muuttaa painamalla nappia eZ430-RF2500T:stä. Datan lähetystiheys voi olla 5 s, 10 s, 20 s, 40 s, 1 m, 2 m tai 4 m. Akkujen ollessa täynnä virran pitäisi riittää noin neljään sataan datalähetykseen. Testausalusta kuluttaa virtaa yhdellä datalähetyksellä 11,1–21,2 mA ja datan vastaanottamisessa 13,3–18,8 mA.

Testausalustan enerchipit ladattiin täyteen ja langaton yhteys muodostettiin tietokoneen ja testausalustan välille. Datan lähetystiheys vaihdettiin 5 sekuntiin ja valokenno peitettiin. Tietokoneelta tarkistettiin, että datalähetys oli 5 sekuntia ja että dataa lähetetään 399 kertaa, jolloin akut ovat täynnä. Sen jälkeen odotettiin akun tyhjenemistä. Neljänsadan datalähetyksen viiden sekunnin välein tulisi kestää noin 33 minuuttia, joten tilanne tarkistettiin 25 minuutin kuluttua konsoli ikkunan avulla ja odotettin virran loppumista.

Ensimmäisellä mittauksella virtaa kesti 36 minuutiksi ja toisella kerralla virta loppui 34 minuutin päästä. Kummallakin kerralla piiri lähetti dataa hieman yli 400 kertaa. Testaus toistettiin vielä neljä kertaa ja virran keston keskiarvoksi sain noin 34 minuuttia ja 23 sekuntia. Testi toistettiin vielä käyttämällä datan lähetystiheytenä kymmentä sekuntia ja testin lopputuloksena tuli noin 400 data-lähetystä. Testien perusteella testausalustan virta riittää valmistajan lupaamaan noin 400:aan lähetyskertaan.

4.4 Volture-energiankerääjän virittäminen

Volture-pietsosähkönkerääjä viritetään tietylle taajuudelle, jolloin kyseisen taajuisesta värinästä saadaan tuotettua mahdollisimman suuri jännite. Taajuusalue, jolle Volture PEH 25w ja SEH 25w voidaan virittää, on 60–140 Hz. Pietsosähkönkerääjä viritettiin 100 Hz:n taajuudelle.

Virittämisessä käytettiin Dactron Comet Shaker -täristintä. Täristimessä on täristinalusta, irrallinen Dactron Comet -etupaneeli ja vahvistin. Täristimen asetukset säädetään tietokoneella olevalla ohjelmalla. Täristintä varten Volturelle suunniteltiin kiinnitysalusta, joka tehtiin Eaglellä. Kiinnitysalustan mitat otettiin täristimen ja Volturen PEH 25w:n ruuvien etäisyyksien perusteella. Kiinnitysalusta oli 159 mm pitkä ja 99,5 mm leveä. Täristintä varten ruuvien kohtien väliset etäisyydet olivat 149 mm pituussuunnassa ja sivuttain etäisyys oli 89,5 mm. Volturen PEH 25w:n kiinnitystä varten ruuvien etäisyydet olivat 85 mm pituussuunnassa ja sivuttain 37,2 mm.

Volturen virittämisessä lisätään painoa Volturen sisältä löytyvän pietsoelementin päähän, jolloin värinästä johtuva liike muuttuu. Virittämisessä käytetään irrotettavissa olevia materiaaleja kuten mehiläisvaha tai teippi. Kun virittäminen on suoritettu, punnitaan lisätty paino ja korvataan se jollakin pysyvällä painolla. Kuvassa 16 on avattu Volture PEH 25w, jossa näkyvälle metallitasolle laitetaan virityksessä lisättävä paino.

KUVA 16. Avattu Volture PEH 25w.

Volturen viritintaajuudeksi valittiin 100 Hz ja säädettiin täristin tärisemään 100 Hz:n taajuudella 20 sekunnin ajan. Tämän 20 sekunnin aikana mitattiin Volturen tuottamaa jännitettä Volturen johdoista. Tämän jälkeen lisättiin tai poistettiin teipin palanen ja tärinä toistettiin, kunnes paino, jolla Volturen tuottama jännite oli korkeimmillaan, oli löydetty. Virityksessä ei otettu huomioon metallilaatan lisäämää painoa.

Värinän vahvuutena käytettiin 1G:tä ja värinän liike huipusta huippuun oli 0,005 mm. Näillä asetuksilla Volturen PEH 25w:stä suurin tuotettu jännite oli 39,1 mV, kun painoa oli lisätty 0,26 grammaa. 0,1 gramman muutos lisätyssä painossa laski tuotetun jännitteen 35 mV:iin. SEH 25w:n valokenno tuotti 2,5–2,9 V:n jännitteen.

5 TULOSTEN YHTEENVETO

Työn ensimmäinen tavoite eli energiankeräysmentelmien kartoitus saavutettiin. Erilaisista energiankeräysmetelmistä käytiin lävitse aurinkoenergia, värinästä saatava energia, lämpöenergiankeräys ja radiotaajuudesta kerättävä energia. Aurinkoenergiasta käytiin lävitse myös valoenergiankeräys sisätiloissa, ja tärinästä saatavasta energiasta käsiteltiin pietsoenergia, sähkömagneettinen energiankeräys ja sähköstaattinen energiankeräys.

Työn toisena tavoitteena oleva testausalustojen vertailu tavoitettiin myös. Vertailua varten kirjoitettiin vertailudokumentti, jossa käsitellään eri valmistajien testausalustoja ja neljä energiankerääjää. Testausalustoista käsitellään laitteiden tekniset tiedot ja testausalustojen perusominaisuuksia käsitellään myös lyhyesti. Vertailusta tehtävä kooste on kirjoitettu vertailudokumentin loppuun.

Työn kolmas tavoite oli demolaitteiston rakentaminen ja testaaminen. Demolaitteistoksi valittiin eZ430-RF2500-SEH -testausalusta ja Volturen PEH 25w- ja SEH 25w -energiankerääjät. Testausalustassa oli valmis demo-ohjelma, jolla testausalustassa olevan valokennon testaukset suoritettiin. Testausalustassa oli charge-pinni, joka oli 1-tilassa valokennon ladatessa piirilevyn akkuja ja 0-tilassa silloin, kun valokenno ei tuottanut virtaa. Eri valonlähteitä testattaessa valokennon toimintaa tarkkailtiin mittaamalla charge-pinnin tilaa yleismittarilla ja valaistuksen vahvuus mitattiin samasta yleismittarista löytyvällä lux-mittarilla. Mittauksissa käytettiin loistelamppua, 40 ja 60 W:n hehkulamppua, 25 W:n energiasäästölamppua, 40 W:n halogeenilamppua ja sinistä led-valaisinta, jossa oli 25 sinistä lediä. Testausalustan valokenno testattiin myös auringosta saatavalla valolla poutasäässä ja sateen aikana. Mittausten perusteella todettiin valokennon toimivan kaikilla testatuilla valonlähteillä.Taulukossa 2 käydään lävitse testausalustan valokennon testauksessa saadut tulokset.

TAULUKKO 2. Valokennon toimivuuden testauksen tulokset.

Valon lähde	**Valaistuksen voimakkuus (lux)**	**Etäisyys valonlähteestä (cm)**	**Valon aallonpituus (nm)**	**Latasiko valokenno akkuja**
Aurinko (poutasäällä)		ulkona auringonpaisteessa		Kyllä
Aurinko (sateella)	n. 370	Sisälle ikkunasta tuleva valo		Kyllä
Loistelamppu	460		630	Kyllä
Hehkulamppu 40 W	350	50	500	Kyllä
Hehkulamppu 60 W	370	100	600	Kyllä
Energiansäästölamppu 25 W	300	30	600	Kyllä
Halogeenilamppu 40 W	1900	150	550	Kyllä
Sininen ledvalaisin	220	20	470	Kyllä

Testausalustan virrankulutusta testattiin lataamalla testausalustan akut täyteen, jonka jälkeen testausalusta suoritti mittauksia testausalustasta löytyvällä lämpötila-anturilla ja lähetti langattoman yhteyden välityksellä tietokoneelle. Lähetystiheytenä oli aluksi 5 sekuntia ja myöhemmin 10 sekuntia. Mittausten perusteella voidaan todeta testausalustan virran kestävän noin 400 datalähetyksen verran,

jonka valmistaja oli luvannut kestoksi. Yhtenä testauksen tavoitteista oli miettiä voiko eZ430-RF2500-SEH-testauslausta käyttää opetuksessa ja jos voi niin missä. EZ430-RF2500-SEH-testausalustaa voi käyttää opetuksessa esimerkkinä langattoman sensoriverkon toiminnasta ja valokennon toiminnasta. Testausalusta toimii myös esimerkkinä mikrokontrolleripiirinä, joka tuottaa oman virtansa.

Pietsosähkönkerääjän virityksessä kiinnitysalusta kiinnitettiin ruuveilla täristimeen ja Volture PEH 25w kiinnitettiin myös ruuvaamalla kiinnitysalustaan, mikä lisää resonanssia ja voi vääristää mittauksia. Tästä syystä jokainen mittaus suoritettiin kolme tai neljä kertaa ennen lisätyn painon muuttamista, jotta mittausvirhettä saataisiin pienennettyä. Viritys tehtiin myös pienellä tärinällä, ja kovemmalla tärinällä olisi saatu tuotettua isommat jännitteet. Testeissä käytetyt pietsoenergiankerääjät eivät ole parhaita mahdollisia testattavia annetun skenaarion kannalta ja tuotettu energia pienenee, kun tärinän aiheuttaa ihmisen liike eikä kone. Testit antavat kuitenkin jonkinlaisen käsityksen siitä, kuinka suurta jännitettä voidaan tuottaa suhteellisen pienellä pietsoenergiankerääjällä.

6 POHDINTA

Työn ensimmäinen tavoite energiankeräysmenetelmien kartoitus oli mielenkiintoinen tehdä, sillä tiesin niistä vain lämpötilaeron ja aurinkoenergian. Normaaleista lampuista kerättävä energia ja sen testaaminen oli myös mielenkiintoista ja lisäsi tietämystä valokennoista paljon. Myös värinästä saatava energia eli pietso-, sähkömagneettinen ja sähköstaattinen energia olivat uusia ja mieluisia aiheita tutkiskella, mutta suurimpana yllätyksenä energiankeräysmenetelmien tutkimisessa oli RF-energian keräämisen mahdollisuus. Sitä olisi ollut mielenkiintoista tutkia, mutta annetun skenaarion eli älyvaatteiden kannalta se ei ollut paras vaihtoehto testata.

Vertailun tekeminen oli vaikeaa suurimmilta osin vertailudokumentin kirjoittamisen takia. Tästä syystä vertailun tekeminen pitkittyi hieman ja viivästytti muun työn valmistumista. Vertailuun olisi ollut hyvä saada useampi vertailtava kohde, varsinkin lämpöenergian keräämisestä ja energiankerääjistä, jotka jäivät neljään Volturen valmistamaan tuotteeseen.

Demolaitteiston rakentaminen ei onnistunut Arduino-alustalle, niin kuin alun perin oli tarkoituksena, mutta siinä saatiin kuitenkin tutkittua energiankeräystä mikrokontrollerilla. Huonona puolena voidaan myös todeta, että laitteistossa oli valmis demo-ohjelma, joten demolaitteistoa ei tarvinnut sen enempää rakentaa. Demoalustan valokennon testauksessa olisi ollut hyvä saada lamppujen tuottaman valkoisen valon aallonpituudet paremmin selville. Lux-mittarin tarkkuutta ei tiedetä, koska se oli yleismittari, jolla oli mahdollista mitata valaistusta, ja sen tarkkuudesta ei ollut varmuutta. Valokennon toimiminen eri lampuilla yllätti, varsinkin LED-valaisimen suhteen, sillä testauksessa käytetty LED-valaisin on tarkoitettu kaapin sisätilan valaisemiseen ja sen tuottama valaistus on pientä. Testauksessa kuitenkin todettiin, että valosta voidaan tuottaa tarpeeksi jännitettä langattoman yhteyden muodostamiseen ja ylläpitämiseen, lämpötila-anturille, datan lähetykseen ja mikrokontrollerin päällä pitämiseen. Ainoa asia, joka jäi vaivaamaan valokennon testauksessa, oli valokennon tuottaman energian määrä, jota en saanut selville.

Pietsosähkönkerääjän virityksessä käytetty Dactronin täristin oli entuudestaan tuntematon laite ja kesti vähän aikaa oppia käyttämään sitä. Pietsosähkönkerääjän virittämisestä olisi voinut ottaa välissä saatuja tuloksia paremmin ylös ja testi olisi ollut hyvä tehdä oskilloskoopilla, jolla olisi voinut saada tulokset suoraan talteen, mutta käytettävissä ei ollut oskilloskooppia, jolla se olisi ollut mahdollista.

Työn tarkoituksena oli tutustua ja kokeilla energiankeräysmenetelmiä ottaen huomioon annettu skenaario sekä rakentaa demolaitteisto, jolla kokeiltiin valittujen energiankeräysmenetelmien toimintaa. Skenaarioksi valittiin älyvaatteissa olevien antureiden virran tuotto yhdellä tai useammalla energiankeräysmenetelmällä. Skenaariota ei toteutettu, vaan sitä käytettiin hyväksi vertailun tekemisessä ja laitteiden valitsemisessa. Annetun skenaarion miettiminen auttoi vertailun tekemisessä, energiankeräysmenetelmän valinnassa sekä eri energiankeräysmenetelmien hyödyllisyyden arvioinnissa käyttötarkoituksen mukaan. Tästä syystä pidänkin käyttötarkoitusta tärkeänä tekijänä oikeaa energiankeräysmenetelmää valittaessa.

LÄHTEET

1. Arduino Duemilanove specification. Saatavissa: http://arduino.cc/en/Main/arduinoBoardDuemilanove. Hakupäivä 14.4.2012

2. About Photovoltaic (PV) and Solar Power Systems. Saatavissa: http://beta.globalspec.com/learnmore/electrical_electronic_components/power_generation_storage/alternative_power_generators/pv_solar_power_supplies. Hakupäivä 24.2.2012.

3. Ilmatieteenlaitos. Valo ja Spektri. Saatavissa: http://www.geo.fmi.fi/oppimateriaali/envisat/valonsade/spektri.html. Hakupäivä 24.9.2012.

4. The Electromagnetic Spectrum. Saatavissa: http://science.hq.nasa.gov/kids/imagers/ems/visible.html. Hakupäivä 24.9.2012.

5. How Do Solar Batteris Work? Saatavissa: http://www.ehow.com/how-does_4597846_solar-batteries-work.html. Hakupäivä 2.3.2012.

6. Fotosähkölaturi. Saatavissa: http://www.tradevv.com/chinasuppliers/rosey1_p_7ed07/china-Solar-mobile-charger-with-high-photoelectric-transformation-rate-long-wroking-life.html. Hakupäivä14.4.2012.

7. Photovoltaic Efficiency-Inherent and System Controlled. Saatavissa: http://www.solar-facts.com/panels/panel-efficiency.php. Hakupäivä 24.2.2012.

8. Beeby, S. P. – Tudor, M. J. – White, N. M. 2006. Energy harvesting vibration sources for Microsystems applications. Saatavissa: http://iopscience.iop.org/0957-0233/17/12/R01/pdf/0957-0233_17_12_R01.pdf. Hakupäivä 14.4.2012.

9. Torres, Erick O. - Rincón-Mora, Gabriel A. Electrostatic Energy Harvester and Li_ion Charger Circuit for Micro-Scale Applications. Saatavissa: http://users.ece.gatech.edu/rincon-mora/publicat/journals/mwscas06_harvestckt.pdf. Hakupäivä 28.2.2012.

10. Waters, Richard L. - Chisum, Brad - Jazo, Hugo - Fralick, Mark 2008. Developmen of an Electro-Magnetic Transducer for Energy Harvesting of Kinetic Energy and its' Applicability to a MEMS-scale Device. Saatavissa: http://www.lumedynetechnologies.com/papers/LTI%20Nanopower%202008%20White%20Paper.pdf. Hakupäivä 28.2.2012.

11. . Mitcheson, Paul David 2005. Analysis and Optimisation of Energy-Harvesting Micro-Generator Systems. Saatavissa: http://cap.ee.ic.ac.uk/~tcg/Microgenerator%20Abstract.pdf. Hakupäivä 14.4.2012.

12. Scansen, Don 2011. Thermoelectric Energy Harvesting. Saatavissa: http://www.digikey.com/us/en/techzone/energy-harvesting/resources/articles/thermoelectric-energy-harvesting.html. Hakupäivä 28.2.2012.

13. Dixon, Bryn. Radio Frequency Energy Harvesting. Saatavissa: http://rfenergyharvesting.com. Hakupäivä 24.2.2012.

14. Donovan, John 2012. Energy Harvesting for Low-Power Wireless Sensor Nodes. Saatavissa: http://www.digikey.com/us/en/techzone/energy-harvesting/resources/articles/energy-harvesting-for-sensor-nodes.html. Hakupäivä 24.2.2012.

15. Piezoelectric materials, Introduction: the piezoelectric effect. Saatavissa: http://www.piezomaterials.com. Hakupäivä 24.2.2012.

16. Types of Solar Energy. Saatavissa: http://www.benefits-of-recycling.com/typesofsolarenergy. Hakupäivä 2.3.2012.

LIITTEET

COMPARISON OF ENERGY HARVESTING KITS AVAILABLE ON THE MARKET

Written by Ilpo Siira

23rd of March 2012, Oulu,Finland

INTRODUCTION

Comparison that is made to choose development boards to test energy harvesting methods. This test is done to see how energy harvesting can work with moving people or in people´s everyday life. Energy harvesting should be able to provide power to wireless links, sending data packets and to power up sensors and/or microcontrollers. Energy harvesting should be done in smart clothing so it should be taken into consideration when choosing the development boards. The first four Volture devices are energy harvesters and not development kits.

Contents

Volture PEH20w (Piezoelectric)

Prize: 399 $ tuned, 299 $ un-tuned.

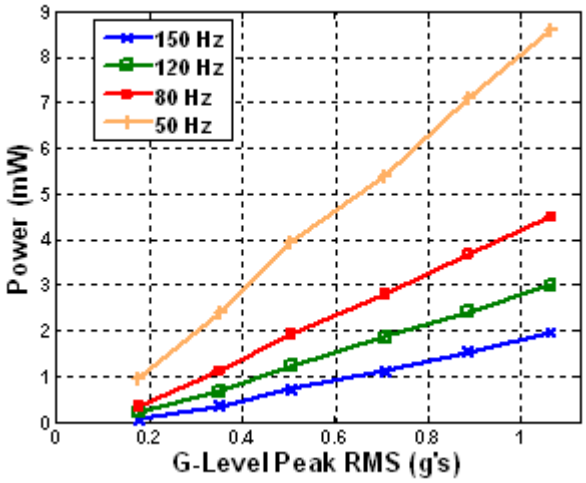
http://www.mide.com/products/volture/volture_catalog.php

Volture PEH 20W and 25W are used to power other applications, where is consistent vibration to be harvested.

Frequency Range (Hz)	80-175
Harvesting Band width (Hz)	3
Device size (in)	3.625 x 1.725 x 0.39
Device wieght (oz)	3
Active elements	1 stack of 2 piezos
Piezo wafer size (in)	1.81 x 1.31 x 0.010
Device capacitance (µF)	0.2
Single Wafer Series Capacitance (nF), measured at 100 Hz	69
Single Wafer Series Resistance (Ohm), measured at 100 Hz	390
Single Wafer Series Capacitance (nF), measured at 120 Hz	69
Single Wafer Series Resistance (Ohm), measured at 120 Hz	340
Max. Tip-to-Tip Displacement (in)	0.10

Generated current not available

The test in the picture is done with 20 kΩ resistance.



This product is actually an energy harvester that can convert vibration to energy and it is best used to power up other applications where consistent vibration can be harvested. The energy harvester has one stack of two piezos and it must be tuned to match the frequency of the vibration source. The frequency range for this product is 80-175Hz. These devices don't work well in harvesting energy from human movement.

Volture PEH25w (piezoelectric)

Prize: 399 $ tuned, 299 $ un-tuned.

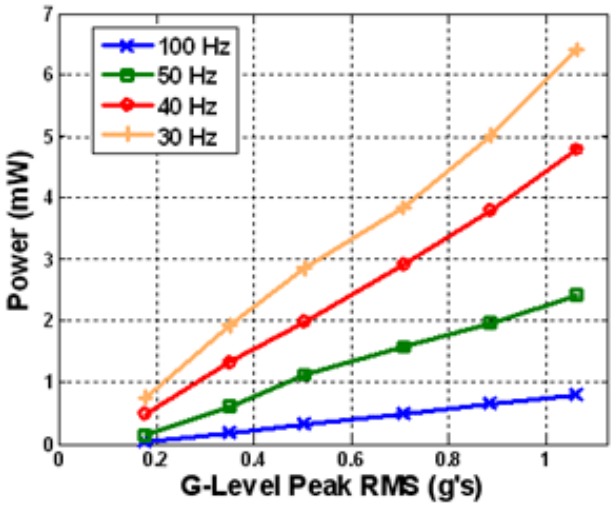
http://www.mide.com/products/volture/volture_catalog.php

Volture PEH 20W and 25W are used to power other applications, where is consistent vibration to be harvested.

Frequency Range (Hz)	60-140
Harvesting bandwidth (Hz)	3
Device size (in)	3.625 x 1.725 x 0.39
device weight (oz)	3
Active elements	1 stack of 2 piezos
Piezo wafer size (in)	1.81 x 1.31 x 0.005
Davice capacitance	0.2
Single Wafer Series Capacitance (nF), measured at 100 Hz	130
Single Wafer Series Resistance (Ohm), measured at 100 Hz	210
Single Wafer Series Capacitance (nF), measured at 120 Hz	130
Single Wafer Series Resistance (Ohm), measured at 120 Hz	175
Max. Tip-to-Tip Displacement (in)	0.15

Generated current not available

The test in the picture is done with 20 kΩ resistance.



This product is actually an energy harvester that can convert vibration to energy and it is best used to power up other applications where consistent vibration can be harvested. The energy harvester has one stack of two piezos and it must be tuned to match the frequency of the vibration source. The frequency range for this product is 60-140Hz. These devices don't work well in harvesting energy from human movement.

Volture SEH20w (piezoelectric & light)

Prize: 499$ tuned & 399 un-tuned

http://www.mide.com/products/volture/volture_catalog.php

Volture SEH20w is an energy harvester that harvests energy from light and vibration. It has one stack of two piezos and the frequency range that the piezo can be tuned in is 75-175Hz. It has a Solarbotics SCC3766 Solar Cell and its short circuit current is 44mA.

Volture SEH25w (piezoelectric & light)

Prize: 499$ tuned & 399 un-tuned

http://www.mide.com/products/volture/volture_catalog.php

Volture SEH25w is an energy harvester that harvests energy from light and vibration. It has one stack of two piezos and the frequency range that the piezo can be tuned in is 60-140Hz. It has a Solarbotics SCC3766 Solar Cell and its short circuit current is 44mA.

EH-Link Energy Harvesting Wireless Sensor Node (EH-Link Pioneer Kit)

Prize: 1 995 $

http://www.microstrain.com/pioneer-kit

Energy harvester inputs	1)5-20V peak AC or DC (piezoelectric, electrodynamic, photovoltaic, electromagnetic) 2) 20 -130 VAC (pulsed piezoelectric) 3) 20-600 mVDC (thermoelectric, peltier,

	thermopile)
Embedded sensors	Onboard triaxial accelerometer, relative humidity and temperature sensor, external single channel differential (wheatstone bridge) input
DC bridge excitation	regulated +3.0 volts DC at 50 mA maximum (pulsed to sensors)
Analog to digital (A/D) converter	12 bit
Energy use	Startup: 12 µJ; Measurement mode: accel – 105 µJ/measurement, relative humidity sensor – 105 µJ/measurement, wheatstone bridge – 168 µJ/measurement; Data transmission: 92.4 µJ/packet
Operating temperature	-20˚C to +60˚C
Dimensions/weight	Standard configurations: 88mm x 39mm x 16mm, 26grams
Software	Node Commander: Windows XP/Vista/7 compatible
Sample rate stability	+/- 3 ppm

The EH-Link pioneer kit has an EH-Link wireless sensor node, 1 solar demo harvester and 1 thermal demo harvester. The EH-Link is versatile and it is designed to operate as a part of MicroStrain's 802.15.4 wireless sensor network.

EH-Link has three different energy harvester inputs: Wide range voltage, that is 5-20V peak AC or DC and it is for energy harvester including piezoelectric, electrodynamic, photovoltaic and electromagnetic. The second energy harvest-

er input is capacitive discharge voltage, which is 20-130 VAC of pulsed piezoelectric. The third energy harvester input is ultralow voltage (20-600 mVDC thermoelectric, peltier, thermopile).

AmbioMote wireless Energy Harvester Kit (piezoelectric)

Prize: 848.50$

http://www.ambiosystems.com/index.php/Development-kits/Combo/Detailed-product-flyer.html

Generated current not available

Key features of AmbioMote24 include:

- Tight integration of energy harvesting electronics and power management with wireless sensor interfaces provides a unique platform for self-powered applications. AmbioMote can accumulate electrical energy provided by an energy harvester until it stores enough for safe completion of a sensing and transmission operation. A battery backup can be used for applications requiring high reliability.
- Intelligent control of energy conversion from energy harvesters provides energy gain of up to 400% (for certain applications).
- Wide range of input voltages (4V-80V) enables operation from a variety of sources such as electromagnetic or piezoelectric energy harvesters.
- AmbioMote24 can use battery power as a back-up or primary source. Ultra-low power consumption of AmbioMote24 can be used to significantly prolong battery life.
- Energy requirements of a specific application are addressed by a multi-bank low-leakage storage. Multi-bank operation allows sizing of energy capacity to various operations and fast initial startup of the sensor node.

- Each sensor node has a unique ID and therefore can be easily identified. Star network architecture enables easy configuration and management of the sensor nodes
- High-speed (2Mbps) wireless link saves energy on data transmissions and enables real-time transmission of data with much higher sampling rates then IEEE802.15.4 (Zigbee) sensor nodes. AmbioMote24 operates in the 2.4 ISM frequency band and therefore can be utilized worldwide. Two hundred and fifty six possible transmission channels enable space sharing by multiple AmbioMote24 networks and coexistence with WiFi devices.
- Long transmission range (80m with a rubber duck antenna) provides wide area coverage.
- A variety of analog (3ch ADC) and digital (digital interrupt-driven I/O, SPI, I2C and serial) interfaces ensures wide compatibility with industry-standard sensors

The development kit contains 4 AmbioMote24-A's. In these 4 AmbioMote24's are an analog temperature sensor, USB interface board, 3D acceleration sensor and an ambient light sensor. All of these are preconfigured so the user doesn't have to configure them. This kit also has one piezoelectric energy harvester in it and user provided energy harvesters can also be used with this kit. This kit seems to have a few sensors, a receiver and piezoelectric energy harvester in it.

Powercast P2110 Energy Harvesting Development Kit for Wireless Sensors

(RF energy)

Prize 1 250 $

http://www.microchipdirect.com/ProductSearch.aspx?keywords=tpwr001

Item	**Description**
Power and Data Transmitter (TX91501-3W-ID)	3-watt, 915 MHz transmitter for power and data with integrated 8 dBi antenna and two power jacks. Sends a pre-programmed transmitter ID that is received by the P2110 Powerharvester component and is decoded by the MCU on the Wireless Sensor Board
P2110 Evaluation Board (P2110-EVB)	Evaluation board (Rev. B) for P2110 Powerharvester Receiver. This board has an SMA connector to connect the antennas and a 10-pin connector for Wireless Sensor Board
Patch antenna	915 MHz directional antenna with 120-degree reception pattern (included with the P2110 evaluation board)
Dipole antenna	915 MHz omni-directional antenna with 360-degree reception (included with the P2110 evaluation board)
Wireless Sensor Board (WSN-EVAL-01)	Sensor for Temp/Humidity/Light and an external input. Plugs into P2110 evaluation board, and sends data to the access point (Microchip 16-bit XLP Development Board).
Microchip 16-bit XLP Development Board (DM240311)	Development platform featuring Microchip´s PIC24F MCU that is pre-programmed to operate as an access point for receiving data from the Wireless Sensor Boards.
Microchip MRF24J40 PICtail/PICtail Plus Daughter Board (AC164134-1)	2.4 GHz, IEEE 802.15.4 radio that plugs into the 16-bit XLP Development Board for receiving data from the Wireless Sensor Boards.
PICkit™ 3 programmer / debugger (PG164130)	Programming tool for updating code on the Wireless Sensor Boards and the 16-bit XLP

	Development Board

Output current up to 50mA.

What's included

- Powercaster™ Transmitter – 915MHz, 3W EIRP
- Powercast P2110-EVB evaluation board, including two antennas and expansion connector (Qty. 2)
- Wireless sensor module - preloaded with operating software (Qty. 2)
- XLP 16-bit Development Board (access point) - pre-loaded with operating software
- MRF24J40MA PICtail Daughter Board - 802.15.4, 2.4 GHz radio module
- PICkit 3 Programmer/Debugger
- USB cables
- Instructions

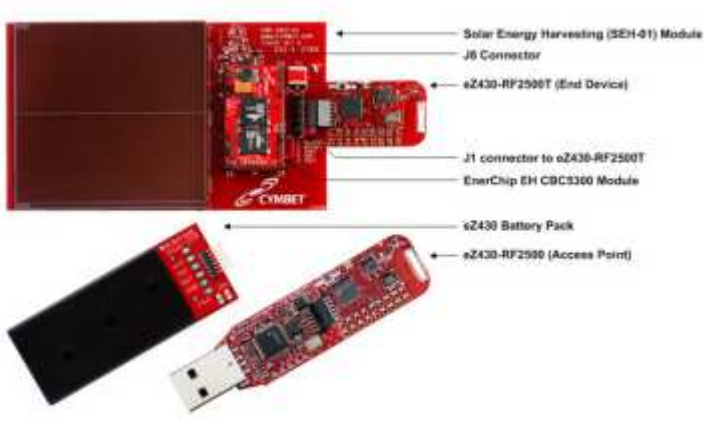
P2110 energy harvesting development kit is powered by RF energy. The transmitter sends a RF-signal that has a frequency of 915 MHz. The receiver receives this signal and the harvester converts the RF energy to DC power. The frequency range that this kit works with is 850-950 MHz.

The power from RF energy should be enough to power the device so it doesn't need batteries. There are some onboard sensors that are also powered by RF energy in this kit and the sensors include temperature, humidity, light and a terminal for an external sensor.

eZ430-RF2500-SEH Solar Energy Harvesting Development Tool (solar/light)

Prize: 149.00 $

http://www.ti.com/tool/ez430-rf2500-seh



PARAMETER	CONDITION	MIN	TYP	MAX	UNIT
Input luminous intensity(1)	Minimum operating lux	200			lux
	Full charge rate	700			lux
Parasitic load current	Boost converter off		800		nA
	Boost converter on		20		µA
Average output power (measured at V_{OUT2} pin)	1000 lux (FL), battery not charging		350		µW
	200 lux (FL), Battery not charging		80		µW
Output voltage, V_{OUT2}	2 µA load, Battery charged	3.5	3.55	3.6	V
V_{BAT} charging voltage			4.06		V
Battery cutoff voltage		3 3.6	3.3		V
Pulse discharge current	20 ms		30		mA
Self discharge rate (non-recoverable av-	25˚C, Per year		2.5		%

erage)				
Operating temperature			0 25 70	˚C
Recharge cycles (to 80% of rated capacity, 4.1V charge voltage)	25˚C	10% depth of discharge	5000	
		50% depth of discharge	1000	
	40˚C	10% depth of discharge	2500	
		50% depth of discharge	500	
Recharge time (to 80% of rated capacity)		4.1V constant voltage	500	min
Capacity		8 µA; 25˚C	100	µAh

(1) Fluorescent (FL) light conditions specifications subject to change without notice.

OS: Microsoft Windows XP,(Vista (32-bit)

Current version: v1.0

Version date: 19-JAN-2009

Pulse discharge current: 30mA

What's Included

- Two eZ430-RF2500T wireless target boards
- One eZ430-RF USB debugging interface
- One AAA battery pack with expansion board (batteries included)
- One SEH-01 Solar Energy Harvester Board
- One MSP430 Development Tool CD containing documentation and development software:
- eZ430-RF2500-SEH Demo and Source Code, SLAC219
- eZ430-RF2500-SEH Development Tool User's Guide, SLAU273
- eZ430-RF2500 Development Tool User's Guide, SLAU227
- MSP430x2xx Family User's Guide, SLAU144

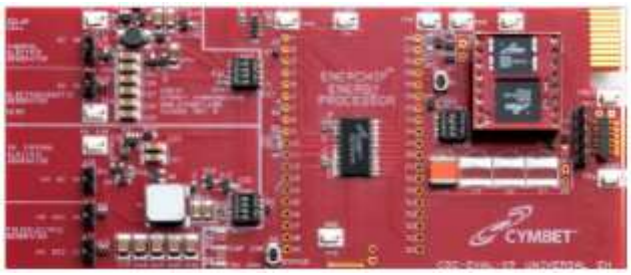
- Code Composer Essentials v3.1 Core Edition, SLAC063

The solar panel in this kit is designed to work indoors, under low-intensity fluorescent lights. It should provide enough power to the application so that it doesn't need batteries. You can also connect another energy harvester to the module. The power harvested by the solar panel will be stored in a pair of thin-film EnerChips. These EnerChips are rechargeable and they will power the applications if there isn't any light to be harvested. This kit has a complete open source application that you can test yourself or modify it to work better for your own energy harvesting project.

CBC-EVAL-09 (solar/light)

Prize: 119.95 $

http://search.digikey.com/us/en/products/CBC-EVAL-09/859-1013-ND/2538181



Parameter	Condition	Min	Typical	Mac	Units
Inout Luminous Intensity (DC In; using PV panel provided)	Minimum operating lux	200 (1)			Lux
	Full charge rate	700 (1)			Lux
Average Continuous Output Power (measured at VOUT pin; using PV panel provided)	1000 Lux (FL), battery not charging		350		µW
	200 Lux (FL), battery not		80		µW

	charging				
DC IN Operating Voltage at Transducer Peak Power Point	source impedance 100Ω to 200Ω	0.2		2.5	V
	source impedance >200Ω	0.5		4.0	V
HV DC IN Operating Voltage at Transducer Peak Power Point	source impedance 10kΩ to 100kΩ	5.0		20	V
DC IN Open Circuit Turn-on Voltage	25˚C	0.4			V
HV DC IN Open Circuit Turn-in Voltage	25˚C	5.0			V
AC IN Operating Voltage at Transducer Peak Power Point	source impedance 100Ω to 10kΩ	1.2		12	Vpp
HV AC IN (1 and 2) Operating Voltage at Transducer Peak Power Point	source impedance 10kΩ to 100kΩ	14.7		57	Vpp
Quiescent Current	Boost converter on		20		µA
	Boost converter off, EnerChips connected		800		nA

		CBC915 off; EnerChips connected		115		nA
VOUT; 2 µA Load		Battery charged	3.5	3.55	3.6	V
VCHG Charging Voltage		25˚C		4.06		V
Battery Cutoff Voltage		4.7kΩ load	3.0	3.3	3.6	V
Pulse Discharge current		20 msec		30		mA
Self-Discharge (non-recoverable average)		25˚C		2.5		% per year
Operating temperature			0	25	70	˚C
Recharge Cycles (to 80% of rated capacity;4.1V charge voltage)	25˚C	10% depth of discharge	5000			
		50% depth of discharge	1000			
	40˚C	10% depth of discharge	2500			
		50 % depth of discharge	500			
Recharge Time (to 80% rated 100µAh capacity)(2)		From 50% state of charge		10		minutes
		From deep discharge		50		minutes

Capacity	100µA discharge; 25˚C		100		µAh

(1) Fluorescent (FL) light.

(2) Assuming charge rate is not limited by input power available from transducer.

Category: Programmers, Development systems

Family: Eval and Demo Boards and Kits

Series: EnerChip

Main Purpose: Power Management, Energy Harvesting

Embedded: No

Utilized IC / Part: CBC915-ACA, CBC51100

Primary Attributes: EM/RF, Solar, Thermal, Vibration Energy Processor

Supplied Contents: Board, Cable, CD, Solar Cell

Pulse discharge current: 30mA

The evaluation kit has a battery module that has two 50µAh solid state batteries connected in parallel in it. These batteries are charged by a solar cell or by another energy harvester should you decide to use your own. The evaluation kit should work with almost all energy harvesting transducers, but the specification of the transducer, like output impedance, has an effect to the amount of power harvested. So the energy harvesting method isn't limited to solar only. Instead you can use RF, thermal or vibration to harvest energy for the applications.

XLP 16-bit Energy Harvesting Development Kit (solar/light)

Prize 195$

http://www.microchipdirect.com/ProductSearch.aspx?keywords=DV164133

Parameters for Cymbet Eval-08 (CBC-Eval-08) are almost the same as CBC-Eval-09.

CBC-Eval-08 has a pulse discharge current of 30mA.

Features

- Solar Energy Harvester with EnerChip storage devices providing backup power
- Flexible development platform with temperature sensors, EEPROM, potentiometer, capacitive sensors, buttons, LEDs, USB, and low power oscillator.
- Expansion connector for development with PICtails such as RF
- Prototyping area for adding additional sensors and circuits
- PICkit 3 Programmer/Debugger for application software development
- PIC24F eXtreme Low Power MCU with 20nA sleep currents

What's included

- Cymbet Eval-08 Solar Energy Harvesting Board and Cable
- XLP 16-bit Development Board
- PIC24F16KA102 Microcontroller
- PICkit 3 Programmer/Debugger
- USB Cables
- XLP 16-bit Energy Harvesting Development Kit user guide
- Quick Start Guide

This kit uses Cymbet's EVAL-08 solar energy harvester to power the kit. The solar energy harvester can harvest energy from indoor and outdoor light. The energy harvested is stored in two thin-film rechargeable energy storage devices, that will power the device if there isn't any ambient energy to be harvested.

The development board has on-board temperature sensors, potentiometer, LEDs, etc. It also has an expansion for PICtail™ modules. These modules can be for example RF or speech playback. Other PICtails modules are also supported.

IPS-EVAL-EH-01: MEC/Energy Harvesting Evaluation Kit (solar/light)

Prize: 129$

http://components.arrow.com/part/detail/51346497S8953917N6341

This kit incorporates a THINERGY® MEC201-7S (0.7 mAh capacity) device, the MAX17710 Energy Harvesting Power Management IC (PMIC) from Maxim Integrated Products, and an amorphous silicon solar panel from Sanyo/Amorton. This kit will also connect to any external (user provided) energy harvester up to 150 mW.

The regulator supports two different output modes:

1) High-Current mode (150mA typical load current)

2) Low-Current mode (100µA typical load current)

The regulator can be configured for 3.3V, 2.3V, or 1.8V operation.

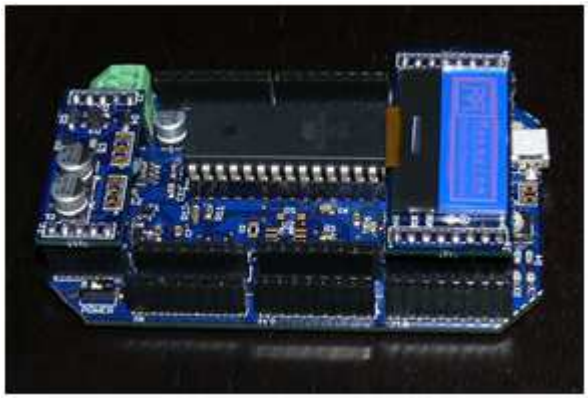
- Autonomous Energy Harvesting & Storage Evaluation Platform for Microelectronics

- Integrated THINERGY® solid-state, rechargeable, Micro Energy Cell for efficient energy storage
- Integrated MAX17710 Power Management IC to manage energy harvesting and storage
- Accept external AC & DC charge sources up to 5V & 150mW
- Integrated boost Converter enables low voltage energy sources >700mV as charge sources
- Unique low energy output regulation mode reduces active quiescent currents to <75nA for output currents <100µA

This evaluation kit powers itself with a silicon solar panel and a micro energy cell for energy storage. The board can be connected into another system, such as a microcontroller or a wireless demonstration kit. You can also add your own energy harvester up to 150 mW to this kit.

Mosquino (an Arduino-based energy harvesting development board)

http://tim.cexx.org/?page_id=760



Important specs:

- Operating voltage range: 1.8 ~3.3V
- Input voltage range: Depends on power shield (the board expects to receive 1.8 ~3.6V pre-regulated or 0 ~6V unregulated from the power shield).
- Speed: 4MHz from onboard crystal oscillator (Atmega´s max. rated speed at 1.8V; you can change to a faster one if operating exclusively at 3.3V).
- Shield pinout: Does not match official Arduino (Duemilanove/etc.) pinout. Instead, the headers have been aligned on a 100-mil grid for compatibility with perfboard and breadboards.

Features:

- Works with the Arduino toolchain or stand-alone AVR toolchains (avr-gcc and avrdude, etc.)
- 3.3V (1.8-3.3V) board designed from the ground up with low power consumption in mind. Target quiescent power draw is <1µA including a realtime clock.
- Power supply (beyond basic battery/USB) implemented as interchangeable shields, allowing harvesting from many power sources (solar, piezo, thermal, RF, …). Power shields may provide regulated or unregulated power, and an aux. output for e.g. clock/memory backup.
- Direct-wire terminal for power input.
- Large, 40-pin ATMEGA644P (Sanguino) used to provide additional I/O. This is important, as the design uses up a couple for power-saving features.
- New shield interface allows up to 3 shields to be installed at once. Common signals (PWR/GND; SPI/I2C buses) are pinned out for each shield block.
- Prioritized power supply allows regulated/unregulated or USB power to be used/connected at any time, without backdriving/damaging each other or sacrificing power efficiency. USB power will supercede battery/etc., saving battery power when other sources are present.

Low-power features:

- PC interface (FTD1232 USB-to-serial, like on Duemilanove) operates from isolated USB supply; does not consume current when USB unplugged. Likewise, bootloader immediately exits if USB cable unplugged.
- Feedback signals to ATMEGA / user program: BUS_SENSE (detect if USB link / “infinite” power is present), POWER_GOOD /changes to false when power source is nearly depleted, if supported by power shield).
- Hardware Real-Time Clock and three hardware INTerrup lines (one for RTC, two uncommitted) to facilitate / encourage event-driven code, CPU sleep mode usage (as opposed to software delay loops) with periodic/scheduled wakeups.
- Rather than using “diode-or”ing, the power supply uses reversed MOSFETs as diodes to isolate the different supplies while avoiding the voltage drop of typical diodes, and cut-off less favorable supplies (e.g. battery) when a more favorable one is available.
- Integrated load-switching prevents the circuit from drawing power until the voltage exceeds minimum requirements and there´s enough power to do something useful with. Since many devices behave strangely and often draw excessive current when driven with too-low voltages, the load switch avoids the problem of a slowly-charging power source never getting “over the hump”.

Mosquino is designed to work with Arduino, but the device isn’t ready yet so it is not commercially available. There seems to be some boards that are working

and the site seems to have links where you can download Arduino 0020 core/libraries and EAGLE schematics, PCB layout and parts list for the main board. Power shields and I/O shields are currently designed and therefore not yet available.

I’m not certain if the person making Mosquino is still working on it, since the web-page doesn’t show when it was updated the last time. If you decide to make your own board you need your own energy harvester as well. Ambient energy harvesters such as thermal, light and vibration should work with it.

SUMMARY

Name	Energy Harvesting Methods	Price	Additional Info
Volture PEH20w & **Volture PEH25w**	piezoelectric	399$ tuned, 299$ un-tuned	Volture PEH20w & 25w are energy harvesters not development or evaluation kits.
Volture SEH20w & **Volture SEH25w**	piezoelectric & light	499$ tuned, 399$ un-tuned	Volture SEH20w & 25w are energy harvesters not development or evaluation kits.
EH-Link Energy Harvesting Wireless Sensor Node (EH-Link Pioneer Kit)	Standard Voltage input: piezoelectric, inductive, EM field, AC or DC sources that produce > 5V instantaneous open circuit voltage. Ultralow Voltage input: thermoelectric generators, low voltage solar cells	1 995$	The kit only has a solar demo board for energy harvesting. If you want to use other energy harvesting methods they need to be bought separately.
AmbioMote wireless Energy Har-	piezoelectric	848.5$	The selling site shows piezoelectric

vester Kit (piezoelectric)			energy harvesters that costs 99$.
Powercast P2110 Energy Harvesting Development Kit for wireless Sensors	RF energy harvesting	1 250$	Has XLP 16-bit Development Board (access point), that is pre-loaded with operating software
eZ430-RF2500-SEH Solar Energy Harvesting Development Tool (Solar/light)	Solar / light	149$	Fluorescent light conditions specification subject to change without notice.
CBC-EVAL-09	solar / light	119,95$	Assuming charge rate is not limited by input power available from transducer.
XLP 16-bit Energy Harvesting Development Kit (Solar/light)	solar / light	195$	Has a Cymbet Eval-08 Solar Energy Harvesting board and Cable and XLP 16-bit Development Board.
IPS-EVAL-EH-01: MEC/Energy Harvesting Evaluation Kit (solar/light)	solar / light	129$	Can connect to any external (user provided) energy harvester up to 150 mW.
Mosquino (an Arduino-based energy harvesting development	information not available	not yet available / boards can be made by getting the schematics	The only Arduino-based energy harvesting development board I was

board)		from site	able to find.

The energy harvesting method of the development kit has a huge effect on the price and the cheapest development kits seems to be the ones using solar / photovoltaic energy harvesting and they cost normally around 100-200$. Thermal development kits are hard to find since everyone seems to sell only sensors. I only found one thermal energy harvesting development kit and those are not made anymore. Piezoelectric is also hard to find and the ones I found costs around 700-800$. With a lot of searching the only RF energy harvesting development kit I found was the Powercast P2110.

Most of the companies making energy harvesting development kits seem to prefer light energy harvesting. It is definitely the cheapest one, but there doesn't seem to be a lot of difference between different development kits. For example XLP 16-bit energy harvesting development kit has a Cymbet Eval-08 Energy Harvesting Board. It is also known as CBC-EVAL-08 and according to specifications it is almost the same as CBC-EVAL-09. Basically XLP 16-bit Energy Harvesting Development Kit may give you more options to try out, but is impossible to be sure of before trying.

There is one development kit coming that is said to work with arduino, but it is not yet available. The device that seemed really good according to specification and can harvest energy with different methods is EH-LINK Energy Harvesting Wireless Sensor Node, but it has a huge setback called the price, as it costs 1 995$.

Most of the development kits in this comparison have inputs for an external energy harvester, but the energy harvesters that I found didn't have much information about them so I decided to leave them out of this comparison.

AmbioMote Wireless energy harvester kit doesn’t have an energy harvester in the kit, but the site suggests a piezoelectric energy harvester that works with the AmbioMote and they claim that the piezoelectric element can continuously drive most of the sensors offered with AmbioMote development kit.

For the Arduino-based development kits I was only able to find Mosquino that isn’t commercially available yet. The reason for it not being available yet is that they still want to make a couple of improvements and there is a note on the site claiming that the Mosquino has not been exhaustively tested yet.

Volture PEH and SEH are energy harvesters and would be a good addition to the development kits that have inputs for other energy harvesters. The development kits that can have another energy harvester are eZ430, CBC-EVAL-09, IPS-EVAL-EH-01 and AmbioMote wireless energy harvester kit. EH-Link is said to operate as a part of MicroStrains wireless network, which has made me a bit worried about it. And as far as testing for the given scenario goes I think that the EH-Link is the most expensive option available.For the scenario given I think that EH-Link may just be too much. The AmbioMote is pre-configured and to use it you don’t have to write any code at all. The product seems to be made with four AmbioMote24-A boards that seem to be relatively small and have sensors in them. This would be a good kit to test according to the scenario in the beginning of this text.

Powercast P2110 is a good option if you are interested in RF energy harvesting and want to see how it actually works. It would also be interesting to see how changing the frequency affect the amount of energy harvested. IPS-EVAL-EH-01 doesn’t have much information about it so I at least would like to have some additional information before I would order it. As far as the solar energy harvesting development kits go, eZ430-RF2500-SEH, CBC-EVAL-09 and XLP 16-bit Energy Harvesting Development Kit doesn’t have many differences between them so choosing between them can be hard.

My proposition when the scenario explained in the introduction is taken in to account is eZ430-RF2500-SEH Solar Energy Harvesting Development Tool and Volture SEH energy harvester. The energy harvester can be either of the possi-

bilities (20w or 25w). My choice is eZ430-RF2500-SEH because I can try other energy harvesters with it and it has an open source application for me to test and modify.

Comport.h

```
#ifndef COMPORT_H
#define COMPORT_H
#include "windows.h"
#include "setupapi.h"
#include "comportthread.h"
#include "nodedb.h"

#include <QStringList>
#include <QTimer>
#include <QCloseEvent>

class ComPort : public QObject
{
   Q_OBJECT

public:
   enum {CP_ERROR, CP_SUCCESS};

                ComPort(QObject *pParent);
   int Enum(QStringList &portList);
   bool Open(const QString &qsPortName, unsigned int baudeRate = 9600);
   bool Read();
   int write(unsigned char *pBuffer, int length);
   bool Close();
   ComPortThread *getThread(){return &comPortThread; }


signals:
   void dataReceived();
   void emptyPacket();
   void unknownMessage();
                void textChanged(const QString &);
                //void storeData(const QString &);
protected:

   bool IsNumeric(LPCTSTR pszString, BOOL bIgnoreColon);

   /// Comport Thread used to read data.
   ComPortThread comPortThread;

};

#endif
```

Comportcombo.h

```
#ifndef COMPORTCOMBO_H
#define COMPORTCOMBO_H
#include <QComboBox>
```

```
/** \brief QCombobox with list of available COM ports
 *
 * This class is a subclass of QCombobox with some special handling for the
COM ports.
 *
 *
 */
class ComPortCombo : public QComboBox
{
   Q_OBJECT

public:
                ComPortCombo(QWidget *parent = 0);

   bool SetComPortList(const QStringList &pList);

};
#endif
```

Comportthread.h

```
#ifndef COMPORTTHREAD_H
#define COMPORTTHREAD_H
#include "windows.h"

#include <QThread>
#include <QQueue>
#include <QMutex>

/// Start Of Packet
#define CPT_SOP '\r' // 0x0D carriage return

/** \brief Read COM port Thread
 *
 * This class is will provide methods to read data from the COM port
 * in its own thread.
 * Calling to the run() function will start the thread and stop() will end it.
 *
 */
class ComPortThread : public QThread
{
   Q_OBJECT

public:
   enum{PACKET_LENGTH = 62, UART_BUFFER_SIZE = 4096};//xx was 56
57

                ComPortThread();
```

```
    virtual ~ComPortThread();
    bool Init(const QString &qsPortName, unsigned int baudeRate = 9600,  int
maxDataLength = 255);
    void Stop();
    bool initiated() { return bInitiated; }
    int WriteCOMMBytes(void * pData, int length);
    int getPacket(QByteArray **pPacket);

signals:
    void packetReady(void);


protected:
                /// Local data buffer (FIFO, trimmed upon uart access)
    typedef struct {
        unsigned char pData[2 * UART_BUFFER_SIZE];    ///< Data buffer
        int inPos;                              ///< Head position
        int outPos;                              ///< Tail position
        int bufferedCount;                          ///< The number of bytes currently in
the buffer
    } UART_BUFFER;

                /// Local data buffer
    UART_BUFFER uartBuffer;

                /// Handle to the COM-port device.
                HANDLE m_hComDev;
    /// Variable to save error number returned by CreateFile()
    DWORD m_error;
                /// The DCB structure defines the control setting for a serial com-
munications device
                DCB m_dcb;
    /// The maximum length returned by \c GetUartData
    int m_maxDataLength;
    /// Variable to tell if COM port has been initiated.
    bool bInitiated;
    /// Variable used to stop the loop in the run() function.
    volatile bool stopped;
    /// packet FIFO queue where all the packets will be stored.
    QQueue<QByteArray *> packets;
    /// Mutex to protect the packet Queue to make it thread safe.
    QMutex mutex;

    void run();
    bool Close();
    bool GetUartData(int *pLength, BYTE *pData);
    bool PopUartBuffer(int *pLength, BYTE *pData);
    bool SearchForSop(void);
    void TrimUartBuffer();
    int ReadCOMMBytes(void *pData, int length);
```

```
};
#endif
```

Conwindow.h

```
#ifndef CONWINDOW_H
#define CONWINDOW_H
#include <QApplication>
#include <QMainWindow>
#include <QLabel>
#include <QMenu>
#include <QTextEdit>
#include <QComboBox>

#include "iohandler.h"
#include "comportcombo.h"
#include "networkview.h"
#include "settingswindow.h"

#include "sensmonmwnd.h"

QT_BEGIN_NAMESPACE
class QAction;
class QMenu;
class QTextEdit;
QT_END_NAMESPACE

/*class ConWindow : public QMainWindow
{
    Q_OBJECT

public:
    ConWindow();
    ConWindow(const QString &fileName);

protected:
    void closeEvent(QCloseEvent *event);
                void ConWindow::updateConsole(const QString data);
private slots:
    void newFile();
    void open();
    bool save();
    bool saveAs();
    void about();
    void documentWasModified();

private:
    void init();
    void createActions();
```

```
    void createMenus();
    void createToolBars();
    void createStatusBar();
    void readSettings();
    void writeSettings();
    bool maybeSave();
    void loadFile(const QString &fileName);
    bool saveFile(const QString &fileName);
    void setCurrentFile(const QString &fileName);
    QString strippedName(const QString &fullFileName);
    ConWindow *findConWindow(const QString &fileName);

    QTextEdit *textEdit;
    QString curFile;
    bool isUntitled;

    QMenu *fileMenu;
    QMenu *editMenu;
    QMenu *helpMenu;
    QToolBar *fileToolBar;
    QToolBar *editToolBar;
    QAction *newAct;
    QAction *openAct;
    QAction *saveAct;
    QAction *saveAsAct;
    QAction *closeAct;
    QAction *exitAct;
    QAction *cutAct;
    QAction *copyAct;
    QAction *pasteAct;
    QAction *aboutAct;
    QAction *aboutQtAct;
};
*/
class ConWindow : public QMainWindow
{
    Q_OBJECT

public:
    ConWindow();
    ConWindow(const QString &fileName);

                QString conTemp;

                int hub_toggle;
                int selectDeselectToggle;
                int node;

                char node1;
                char node2;
```

```
                char node3;
                char node4;
                char node5;
                char node6;
                char node7;
                char node8;

protected:
    void closeEvent(QCloseEvent *event);
                QMainWindow *tMyParent;
                void ConWindow::updateConsole(const QString data);

public slots:

private slots:
    void newFile();
    void open();
    bool save();
    bool saveAs();
    void about();
    void documentWasModified();
                void storeData(const QString &);
                void dontShowHub();
                void clearWindow();
                void deselectallnodes();
                void testCheck(bool check );
                //void updateConsole();

private:
    void init();
    void createActions();
    void createMenus();
    void createToolBars();
    void createStatusBar();
    void readSettings();
    void writeSettings();
    bool maybeSave();
    void loadFile(const QString &fileName);
    bool saveFile(const QString &fileName);
    void setCurrentFile(const QString &fileName);
    QString strippedName(const QString &fullFileName);
    ConWindow *findConWindow(const QString &fileName);
                void ConWindow::nodeActions();

                QGroupBox *createNonExclusiveGroup();

    QTextEdit *textEdit;
    QString curFile;
    bool isUntitled;
```

```
    QMenu *fileMenu;
    QMenu *editMenu;
    QMenu *helpMenu;
                QMenu       *checkMenu;
    QToolBar *fileToolBar;
    QToolBar *editToolBar;
    QAction *newAct;
    QAction *openAct;
    QAction *saveAct;
    QAction *saveAsAct;
    QAction *closeAct;
    QAction *exitAct;
    QAction *cutAct;
    QAction *copyAct;
    QAction *togglehubAct;
    QAction *aboutAct;
    QAction *aboutQtAct;
                QAction *clearConsole;
                QAction      *checkNode;
                QAction      *node1Act;
                QAction      *node2Act;
                QAction      *node3Act;
                QAction      *node4Act;
                QAction      *node5Act;
                QAction      *node6Act;
                QAction      *node7Act;
                QAction      *node8Act;
                QAction *deselectAllAct;

                QFont modeFont;
};
#endif
```

Edge.h

```
#include <QGraphicsItem>
#include "node.h"


class Node;

class Edge : public QGraphicsItem
{
public:
    Edge(Node *sourceNode, Node *destNode);
    ~Edge();

    Node *sourceNode() const;
    void setSourceNode(Node *node);
```

```
    Node *destNode() const;
    void setDestNode(Node *node);

    void adjust();

    enum { Type = UserType + 2 };
    int type() const { return Type; }

protected:
    QRectF boundingRect() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);

private:
    Node *source, *dest;

    QPointF sourcePoint;
    QPointF destPoint;
    qreal arrowSize;
};

#endif
```

Graphwindow.h

```
#ifndef GRAPHWINDOW_H
#define GRAPHWINDOW_H
#include <QApplication>
#include <QtGui>
#include <QMainWindow>
#include <qlayout.h>
#include <qlabel.h>
#include <qpainter.h>
#include <qwt_plot_layout.h>
#include <qwt_plot_curve.h>
#include <qwt_scale_draw.h>
#include <qwt_scale_widget.h>
#include <qwt_legend.h>
#include <qwt_legend_item.h>


#include <QT>
#include <QStyle>
#include <QMotifStyle>
#include <QPlastiqueStyle>
#include <QCDEStyle>
#include <QCleanlooksStyle>
#include <QMdiArea>
```

```
#include <QtGui/QMdiArea>
#include <QtGui/QMdiSubWindow>
#include <qfont.h>

#include <QString>

#include "nodedb.h"
#include "sensorplot.h"


class GraphWindow : public QMainWindow
{
   Q_OBJECT

public:
   GraphWindow(QObject *pParent);
              void setNodesList(const QStringList &pList);


public slots:
              void addNode(int addr);
              void removeNode(int addr);
              void graphData(const QString &);
              void setCurrentNode(int);

protected:
              QMainWindow *pMyParent;

private:
   void init();
              void getNodeName(int addr, QString &name);

              QWidget vBox;
              QVBoxLayout *layout;
              QComboBox *nodes;
   SensorPlot *plot;

              NodeDb::DATA *node;
};

#endif
```

iohandler.h

```
#ifndef IOHANDLER_H
#define IOHANDLER_H
#include "comport.h"
```

```
#include <QStringList>
#include <QMessageBox>
#include <QMainWindow>

class ConWindow;
/** \brief Handler of all communication with connected device.
 *
 * This class provides methods to give a list of connected devices that are appli-
cable
 * for this application and the methods to communicate with the selected device.
 *
 * This class is implemented with platform independent functionality but for the
actual access
 * to the underlaying hardware the ComPort class is implemented with all the
platform dependent functionality.
 *
 *
 */
class IoHandler : public QObject
{
   Q_OBJECT

public:
   enum{IO_ERROR, IO_SUCCESS};

                IoHandler(QObject *pParent = NULL);
   virtual IoHandler::~IoHandler();
   int RefreshComPort();
   QStringList &getComPortList();

   int openComPort(const QString &portName);
   int closeComPort();
                QString conTemp;
                void Data_available(const QString data);
                void IoHandler::start_con(void);
                // ComPort *comPort;
signals:
   void dataReceived();
   void unknownMessage();
   void emptyPacket();
                void updateConsole(const QString &);
                void textChanged(const QString data);
                void storeData(const QString &);
private slots:
   void handlePacket();

protected:
   ComPort *comPort;
   QMainWindow *pMyParent;
```

```
                ConWindow *conWindow;

    bool m_bComPortListOk;
    QStringList m_lstComPort;

    /// Mutex to protect the Node database to make it thread safe.
    QMutex mutex;
    /// Pointer to the node data base
    NodeDb *pNodeDb;

    void handleData(QByteArray *pPacket);
private:
                QByteArray* id_node;
                QByteArray* id_temperature;
    QByteArray* id_voltage;
                QByteArray* id_strength;
                QByteArray* id_EOP;
                QByteArray* id_time;
                //QByteArray* id_onoff;
                 QTime timeStamp2;


};

#endif
```

networkview.h

```
#ifndef NETWORKVIEW_H
#define NETWORKVIEW_H
#include "qgraphicsview.h"
#include "node.h"
#include "nodedb.h"


class NetworkView : public QGraphicsView
{
    Q_OBJECT

public:
    typedef struct
    {
        Node *pNode;
        Edge *pEdge;
    }LIST_DATA;

    enum{CHECK_ALIVE_INTERVAL=1000};

    NetworkView(QWidget *pParent = NULL);
```

```
    void addNode(int index);
    void connectNodes(int srce, int dest);
    void updateNode(int index);
    void deleteAll();
    void startCheckAliveTimer();
    void stopCheckAliveTimer();
    void changeTempUnit();

signals:
                void nodeAdded(int index);
                void nodeRemoved(int index);


protected:
    void keyPressEvent(QKeyEvent *event);
    void timerEvent(QTimerEvent *event);
    void wheelEvent(QWheelEvent *event);
    void drawBackground(QPainter *painter, const QRectF &rect);
    void deleteNode(int index);

    void scaleView(qreal scaleFactor);

    QGraphicsScene *pScene;

private:
    QWidget *pParentWindow;
    /// Timer ID used for the time between each check to see if a node is alive or
not.
    int timerId;
    /// Timer interval used to determine if Node is alive or not.
    int aliveTimerInterval;

    QList<LIST_DATA *> nodeList;

    QPixmap pxmTiLogo;
    QPixmap pxmTiLogo2;//x
                QPixmap pxmTiLogo3;
                QPixmap pxmTiLogo4;
    NodeDb *pNodeDb;
};

#endif
```

node.h

```
#ifndef NODE_H
#define NODE_H
#include <QGraphicsItem>
```

```
#include <QList>
#include <QTime>
#include <QFont>
#include <QPixmap>
#include <QBrush>

#include <qlcdnumber.h>

#include "edge.h"



class Edge;
class NetworkView;
class QGraphicsSceneMouseEvent;
class NodeDb;



class Node : public QGraphicsItem, public QObject
{
public:
    typedef enum{ACCESS_POINT, END_DEVICE}NODE_TYPE;
    typedef struct
    {
        NODE_TYPE       type;
        int                     addr;
        int                     xPos;
        int                     yPos;
                        int                             strength;
        float           temp;
        float           voltage;
                        int                             re;

                        int                             t_time_on;
                        int                             fadeNumber;
                        int                             fadeTime;
                        int                             newNode;
                        int                             deletedNode;

    }NODE_DATA;

                        QBrush fill[40];
    Node(NetworkView *networkView);
                NODE_DATA nodeData;
    void addEdge(Edge *edge);
    QList<Edge *> edges() const;

    enum { Type = UserType + 1 };
    int type() const { return Type; }
```

```
    bool advance();

    QRectF boundingRect() const;
    QPainterPath shape() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);

    void setNodeData(int index);
    void updateData(int index);
    int getElapsedTime();
    void removeEdge(int childAddr = 0);
    int getAddr() { return nodeData.addr; }
    float getTemp() { return nodeData.temp; }
    int getType() { return nodeData.type; }
    void setTempUnit();
    void setColor(const NODE_TYPE type);
                int Node::find_active_node();
                void Node::current_updateData(int index);
                int timerId () const;

                /*struct colors{
                        int alpha;
                        int r;
                        int g;
                        int b;
                        int newnodeflag;
                        int delnodeflag;
                };
                struct colors node_colors[100];*/


protected:
    QVariant itemChange(GraphicsItemChange change, const QVariant &value);

    void timerEvent(QTimerEvent *);

   // NODE_DATA nodeData;

private:
    QList<Edge *> edgeList;
    QPointF newPos;
    NetworkView *graph;
                QPixmap pixmap,alpha_pix,back_pix,pixmap2,pix;

    QColor nodeColor;
    QColor nodeColorDark;

                QFont modeFont;
                QFont addrFont;
```

```
            QFont tempFont;
            QFont battFont;
    QFont timeFont;
            QFont end_deviceFont;
            QFont       keyFont;

    QTime timeStamp;
            QDate dateStamp;

    NodeDb *pNodeDb;

    // Flag used to highlight node.
    bool updated;
    int updateTimerId,fade_speed,updateTimerId2,updateTimerId3;

    bool celsius;
            qreal arrowSize;

};

#endif
```

nodedb.h

```
#ifndef NODEDB_H
#define NODEDB_H
#include <QQueue>
#include <QMutex>
#include <QFile>

#include "node.h"

#define DB_WRITE(T)
#define DB_OPEN

class NodeDb
{
public:
    typedef enum{DS_NORMAL, DS_NEW, DS_UPDATED,
DS_MOVE}DB_STATUS;
    typedef struct
    {
        int             parent;
                        int             addr;
        Node::NODE_TYPE  type;
        float           temp;
        float           voltage;
                        double          angle;
                        int             prevStrength;
```

```
        int             strength;
                                int             re;

                                int
t_time_on;
                                int             xpos;
        int             ypos;
                                int
                fadeNumber;
                                int
                fadeTime;
                                int
                newNode;
                                int
                deletedNode;
        DB_STATUS       status;
    }DATA;

    NodeDb();
    virtual ~NodeDb();
    bool deleteNode(int index);
    bool deleteAll();
    int addData(DATA *pData);
    bool getData(int index, DATA **pData);
    int getIndex(int addr);
                void updateEndpointPos();

protected:
   // void updateEndpointPos();

private:
                int calculateRadius(int strength);

};

#endif
```

sensormend.h

```
#ifndef SENSMONMWND_H
#define SENSMONMWND_H
#include <QMainWindow>
#include <QLabel>
#include <QMenu>
#include <QTextEdit>
#include <QComboBox>
#include <QAbstractEventDispatcher>

#include "iohandler.h"
```

```
#include "comportcombo.h"
#include "networkview.h"
#include "settingswindow.h"

#include "graphwindow.h"

class NetworkView;
class DeviceChange;
class IoHandler;
class GraphWindow;

QT_BEGIN_NAMESPACE
class QGroupBox;
QT_END_NAMESPACE

/** \brief The Main Window for the Sensor Monitor
 *
 *
 */
class SensMonMWnd : public QMainWindow
{
   Q_OBJECT

public:
   enum{TIMER_DEV_CHANGE = 2000};

   NetworkView *networkView;

   SensMonMWnd();
   static bool wmEventFilter(void *message);
   void setAppPath(char *path);
            //QString IoHandler:conTemp;
            QString conTemp;

public slots:
            void showHelp();
            void showConsole();
            //void updateConsole(QString data);
            void closeConsole();


protected:
   void deviceChange();

   /// Pointer to the node data base
   NodeDb *pNodeDb;
   /// The application path is stored on startup.
   QString appPath;

private slots:
```

```
            void toggleFullscreen();
            void escapeToggle();
            void showAbout();
            void showGraph();
    void showSettings();

    void removeAllNodes();
    void onStartCapture();
    void startCapture(bool devChangeEvent = false);
    void onPauseCapture();
    void onStopCapture();
    void stopCapture(bool devChangeEvent = false);
    void refreshComPort();
            void getNodeList(QStringList &pNodeList);
    void dataReceived(/*int index*/);
    void timeOutDevChange();
    void showHelpInfo();
            void updateConsole(const QString &);

private:
    void createActions();
            void createMenus();
    void createToolBars();
    void createStatusBar();

    void closeEvent(QCloseEvent *event);
    bool eventFilter(QObject *obj, QEvent *event);
            void setConsoleOpen();
            void setConsoleClose();

             //QCheckBox *checkBox1;

    //TextWindow *aboutWindow;
            //QMenu *menu;

    // Toolbar for start and stop
    QToolBar *captureToolBar;
    // Toolbar for the combobox.
    QToolBar *comPortToolBar;
    // Toolbar for the help functions.
    QToolBar *helpToolBar;
    // Put the combobox with COM ports on the toolbar.
    ComPortCombo comPortComboBox;
    /// Flag to indicate if applicable device connected
    bool applDeviceConnected;
    /// Flag to indicate that device change event has occured.
    bool bDevChanged;
    /// Flag to indicate if COM port is open or not
    bool bComPortOpen;
    /// Flag used to control if nodes should be deleted or not after pause or stop.
```

```
    bool bDeleteNodes;
    /// Event handler
    QAbstractEventDispatcher *m_EventDispatcher;

                QShortcut *helpShortcut;
                QShortcut *esc;
                QShortcut *altEnter;

    QAction *fullscreen;
    QAction *stopAction;
    QAction *captureAction;
    QAction *pauseAction;
    QAction *refreshAction;
    QAction *settingsAction;
    QAction *helpAction;
    QAction *aboutAction;
                QAction *graphAction;
                QAction *consoleAction;

    QLabel *connectStatus;

    IoHandler *ioHandler;
                GraphWindow *graph;
                QStringList m_nodeList;

                SettingsWindow *settingsWindow;



};

/** \brief Device Change Event Class
 *
 *  This class is needed to define a user event
 *  that will be used to fetch the WM_DEVICECHANGE message
 *  when a USB device has been connected/disconnected.
 *
 */
class DeviceChangeEvent : public QEvent
{
public :
    enum{DeviceChanged = 1050};
    DeviceChangeEvent(): QEvent((QEvent::Type)DeviceChanged) {}

};

/*
class ConWindow : public QMainWindow
{
    Q_OBJECT
```

```
public:
    ConWindow();
    ConWindow(const QString &fileName);

                QString conTemp;

                int hub_toggle;
                int selectDeselectToggle;
                int node;

                char node1;
                char node2;
                char node3;
                char node4;
                char node5;
                char node6;
                char node7;
                char node8;

protected:
    void closeEvent(QCloseEvent *event);
                QMainWindow *tMyParent;
public slots:



private slots:
    void newFile();
    void open();
    bool save();
    bool saveAs();
    void about();
    void documentWasModified();
                void storeData(const QString &);
                void dontShowHub();
                void clearWindow();
                void deselectallnodes();
                void testCheck(bool check );
                //void updateConsole();
private:
    void init();
    void createActions();
    void createMenus();
    void createToolBars();
    void createStatusBar();
    void readSettings();
    void writeSettings();
    bool maybeSave();
    void loadFile(const QString &fileName);
```

```
    bool saveFile(const QString &fileName);
    void setCurrentFile(const QString &fileName);
    QString strippedName(const QString &fullFileName);
    ConWindow *findConWindow(const QString &fileName);
                void ConWindow::nodeActions();

                QGroupBox *createNonExclusiveGroup();

    QTextEdit *textEdit;
    QString curFile;
    bool isUntitled;

    QMenu *fileMenu;
    QMenu *editMenu;
    QMenu *helpMenu;
                QMenu       *checkMenu;
    QToolBar *fileToolBar;
    QToolBar *editToolBar;
    QAction *newAct;
    QAction *openAct;
    QAction *saveAct;
    QAction *saveAsAct;
    QAction *closeAct;
    QAction *exitAct;
    QAction *cutAct;
    QAction *copyAct;
    QAction *togglehubAct;
    QAction *aboutAct;
    QAction *aboutQtAct;
                QAction *clearConsole;
                QAction      *checkNode;
                QAction      *node1Act;
                QAction      *node2Act;
                QAction      *node3Act;
                QAction      *node4Act;
                QAction      *node5Act;
                QAction      *node6Act;
                QAction      *node7Act;
                QAction      *node8Act;
                QAction *deselectAllAct;

                QFont modeFont;
};
*/
#endif
```

sensorplot.h

```
#ifndef SENSORPLOT_H
#define SENSORPLOT_H

#include <QTime>
#include <qwt_plot.h>
#include "nodedb.h"

#define HISTORY 60 // seconds
#define MAXNODES 8
#define RANGE 10

class QwtPlotCurve;

class SensorPlot : public QwtPlot
{
   Q_OBJECT
public:
   enum CpuData
   {
      Strength,
      Temperature,

      NCpuData
   };


   SensorPlot(QWidget * = 0);
   const QwtPlotCurve *cpuCurve(int id) const
      { return data[id].curve; }
            void graphData(const QString &eventData);
            void updateCurrentNode(int addr);

private slots:
   void showCurve(QwtPlotItem *, bool on);

private:
   /*struct
   {
      QwtPlotCurve *curve;
      double data[HISTORY];
   } data[NCpuData];*/

            typedef struct
   {
      QwtPlotCurve *curve;
      double data[HISTORY];
   } NODEPLOTDATA;

            NODEPLOTDATA data[NCpuData];
```

```
        double timeData[HISTORY];

        int dataCountList[MAXNODES];
        int currentNode;
        QTime prevTime;

        double nodeDataList[MAXNODES][NCpuData][HISTORY];
        double timeDataList[MAXNODES][HISTORY];
        QTime baseTimeList[MAXNODES];

        int yAxisMinList[MAXNODES][NCpuData];
        int yAxisMaxList[MAXNODES][NCpuData];

};

#endif
```

settingswindow.h

```
#ifndef SETTINGSWINDOW_H
#define SETTINGSWINDOW_H
#include <QDialog>

#include "ui_SettingsWindow.h"


class SettingsWindow : public QDialog

{
        Q_OBJECT

public:
   enum{BTN_OK};
        SettingsWindow(QWidget *parent = 0);

public slots:

private slots:
   void accept();
   void toggleTempUnit();
signals:

private:
        Ui_SettingsWindow ui;

   int nodeTimeout;
   bool tempInCelsius;
};
```

```
#endif
```